Inside the Pentium® 4 Processor Micro-architecture

Next Generation IA-32 Micro-architecture

Avi Kumar

Microprocessor Architecture

Performance Manager

Desktop Platforms Group

Contributors: Eric Sprangle, Doug Carmean and Avi Kumar

Intel

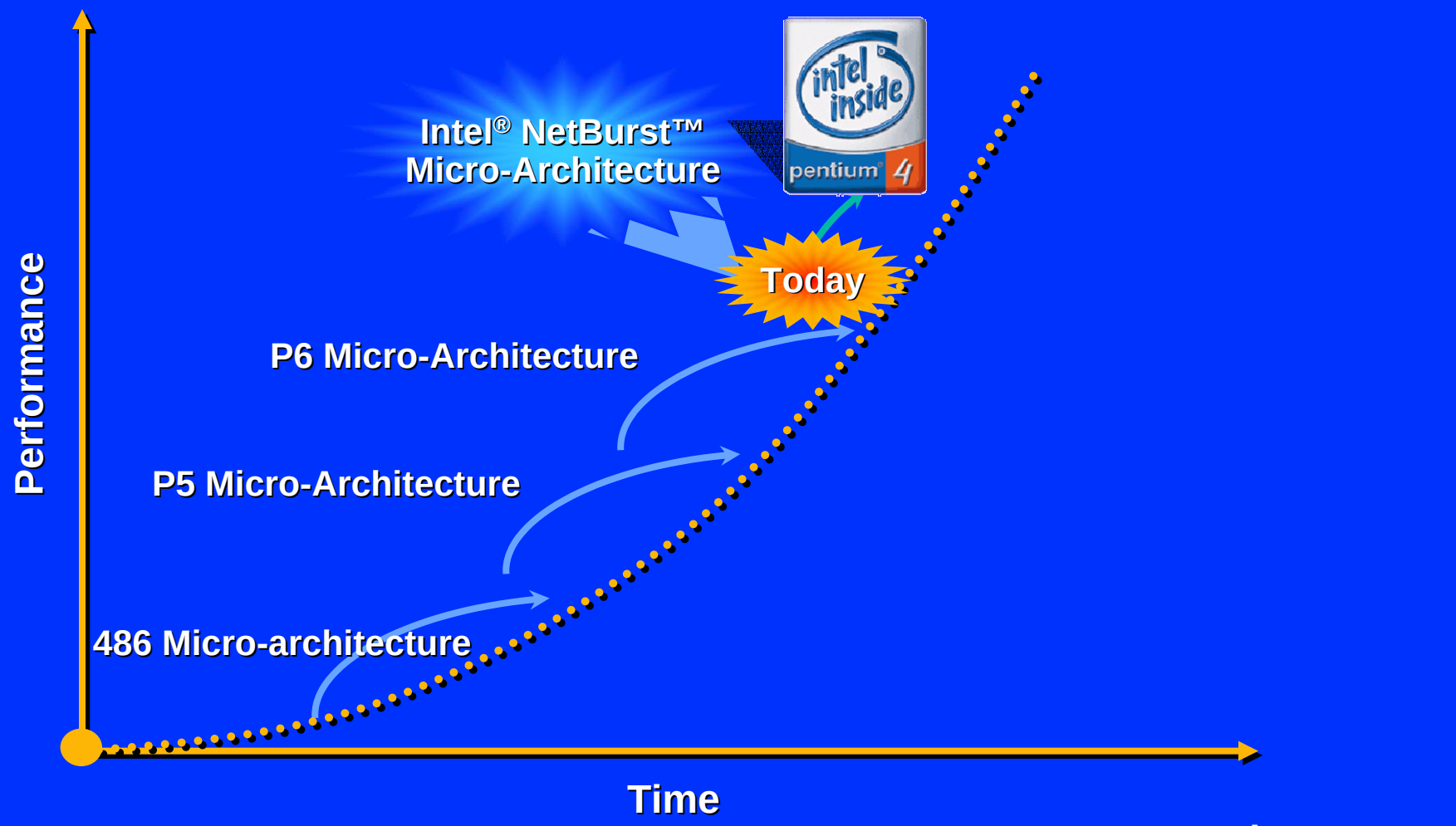
Austin

Agenda

- IA-32 Processor Roadmap
- Design Goals
- Architecture 101
  - Frequency
  - Instructions Per Cycle
- Advanced Architectural Concepts
  - Data Speculation
  - Data Prefetching
  - HyperThreading™
- Summary

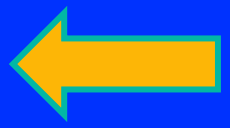
Intel
Austin

Intel® Pentium® 4 Processor



Intel
Austin

Agenda

- IA-32 Processor Roadmap
- Design Goals 
- Architecture 101
  - Frequency
  - Instructions Per Cycle
- Advanced Architectural Concepts
  - Data Speculation
  - Data Prefetching
  - HyperThreading™
- Summary

Intel Austin

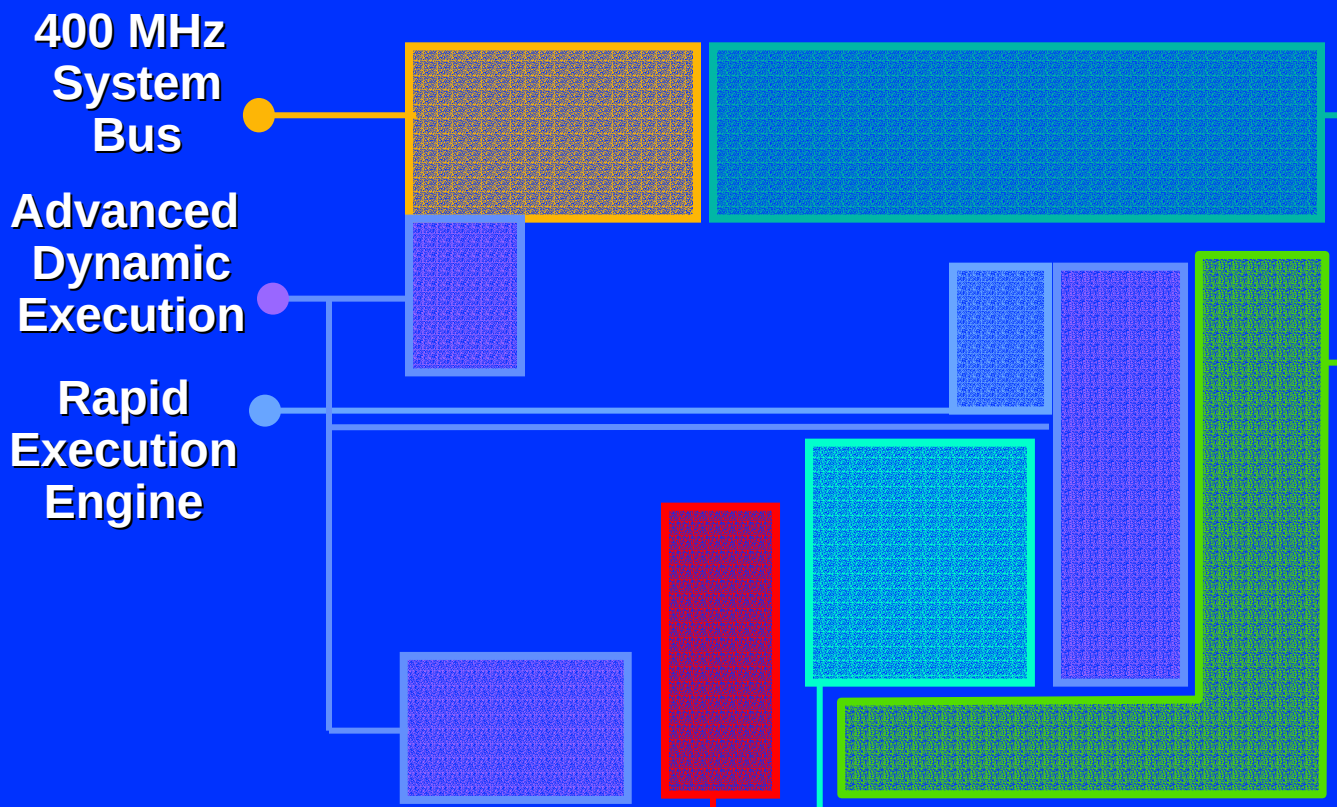
Intel® Pentium® 4 Processor Design Goals

- Deliver highest performance possible on both existing and emerging applications
- Enable performance headroom and scalability for the future

Micro-architecture will be basis for high-end Intel IA-32 processors for the next several years

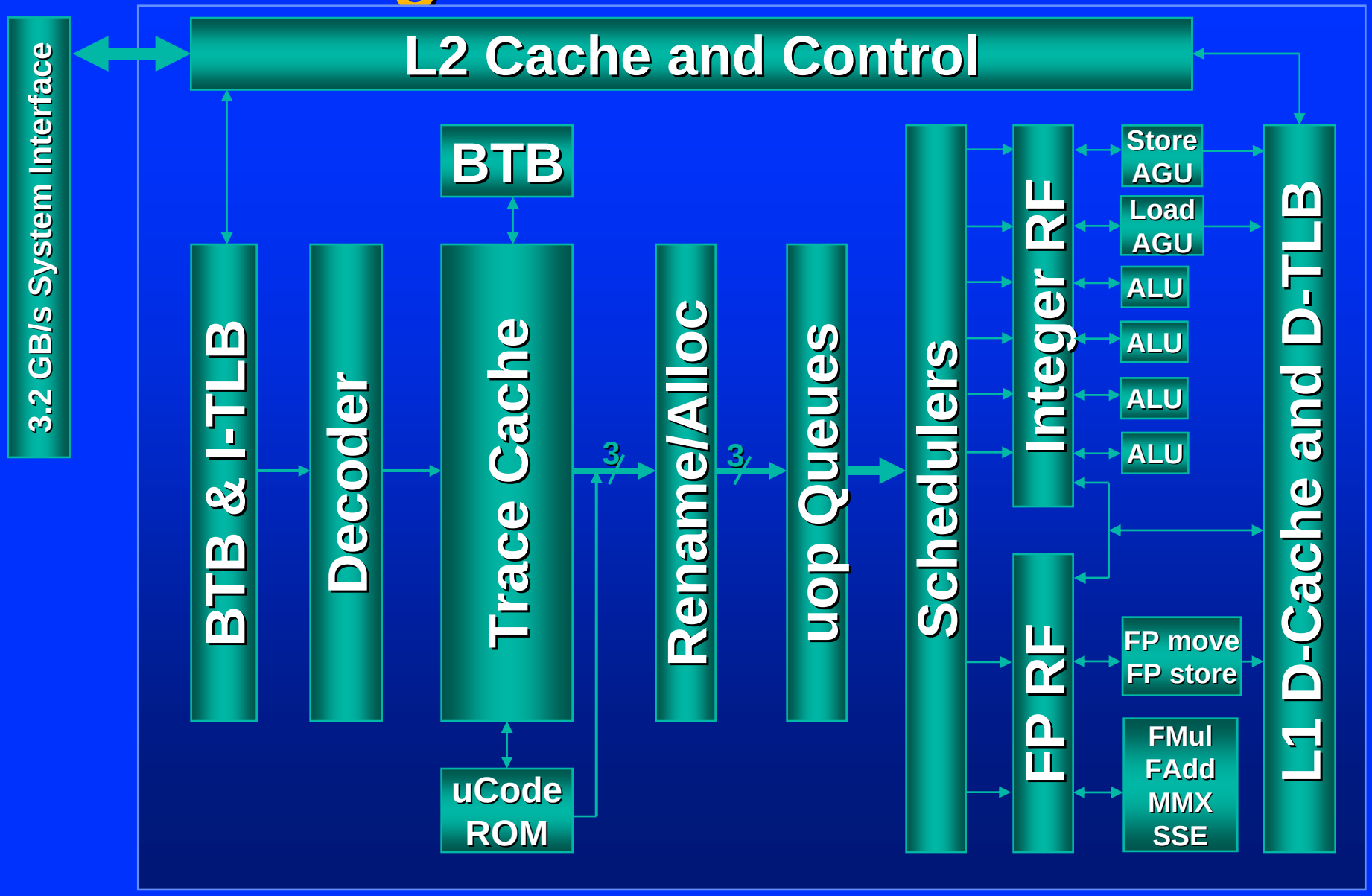
Intel Austin

Intel® Netburst™ Micro-Architecture

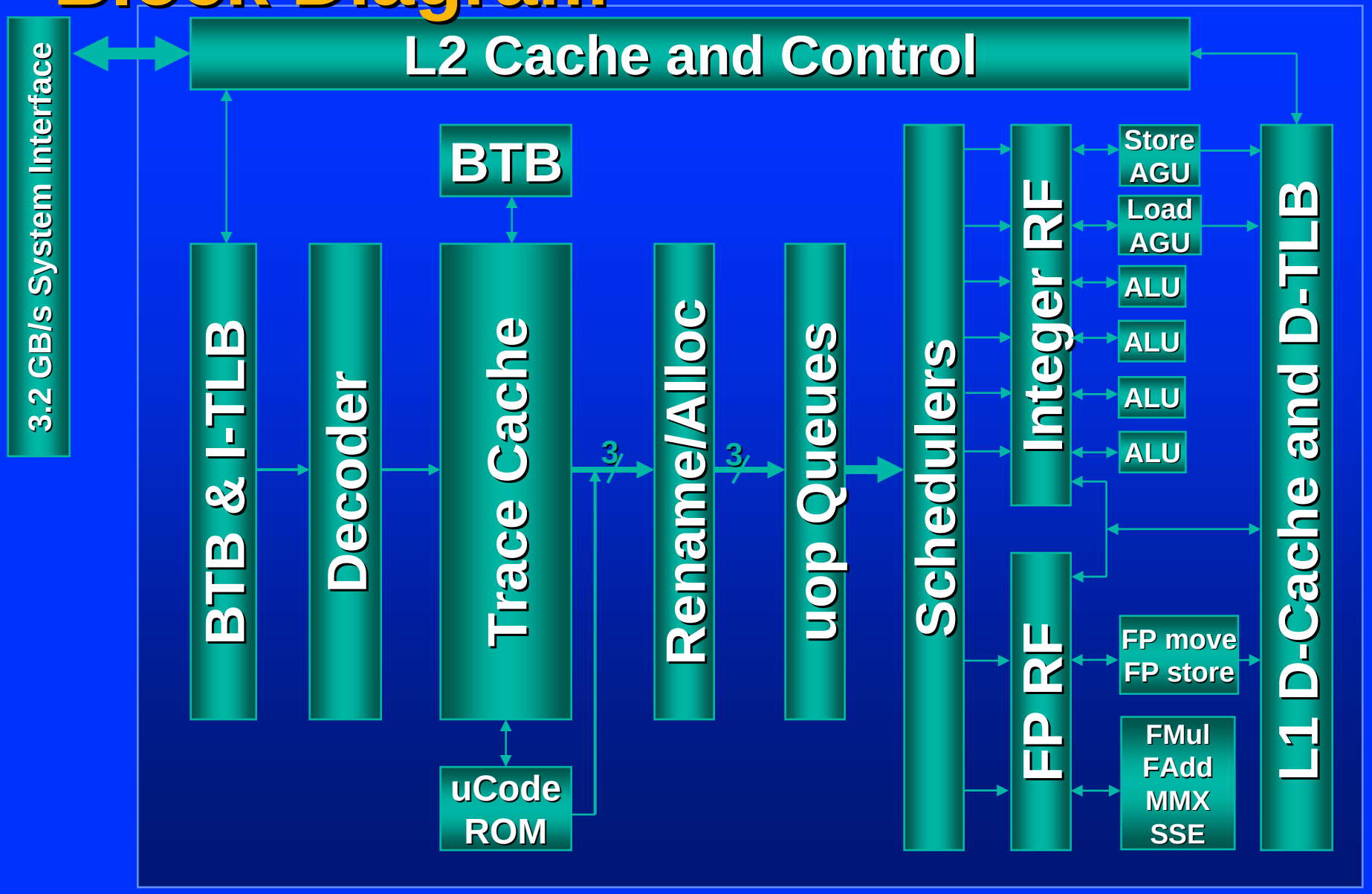


Intel
Austin

Pentium® 4 Processor Block Diagram

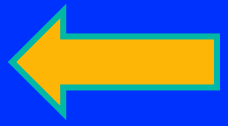


Pentium® 4 Processor Block Diagram



Agenda

- IA-32 Processor Roadmap
- Design Goals
- Architecture 101
  - Frequency
  - Instructions Per Cycle
- Advanced Architectural Concepts
  - Data Speculation
  - HyperThreading™
- Summary



Intel Austin

CPU Architecture 101

Intel
Austin

Frequency

What limits frequency?

- Process technology
- Microarchitecture

- On a given process technology
  - Fewer gates per pipeline stage will deliver higher frequency

Frequency is driven by Microarchitecture

Intel Austin

Pentium 4 vs P6

Basic P6 Pipeline

Intro at 733MHz .18µ

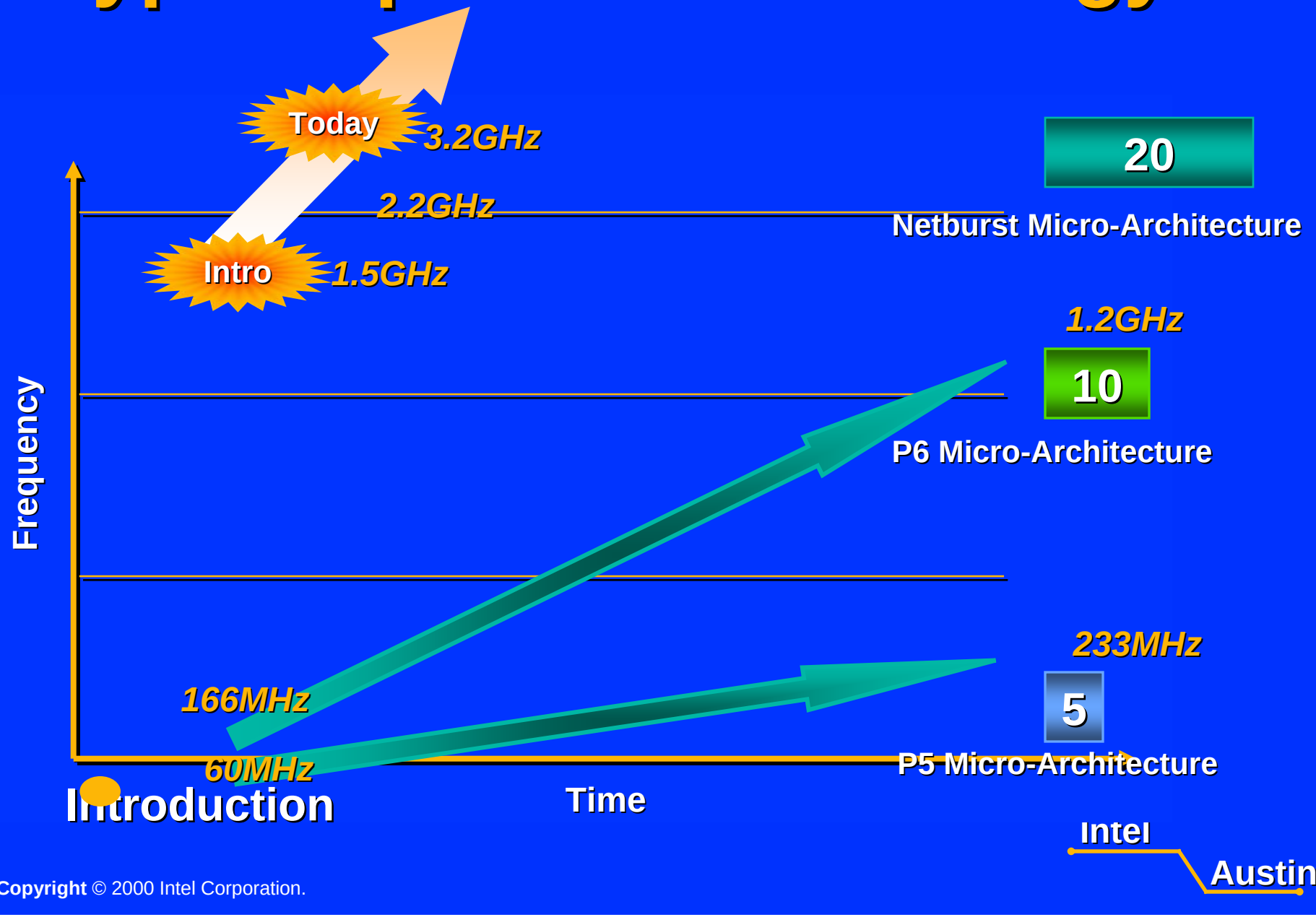
Basic Pentium® 4 Processor Pi

Intro at ≥ 1.4GHz .18µ

Hyper Pipelined Technology enables industry leading performance and clock rate

Austin

Hyper Pipelined Technology



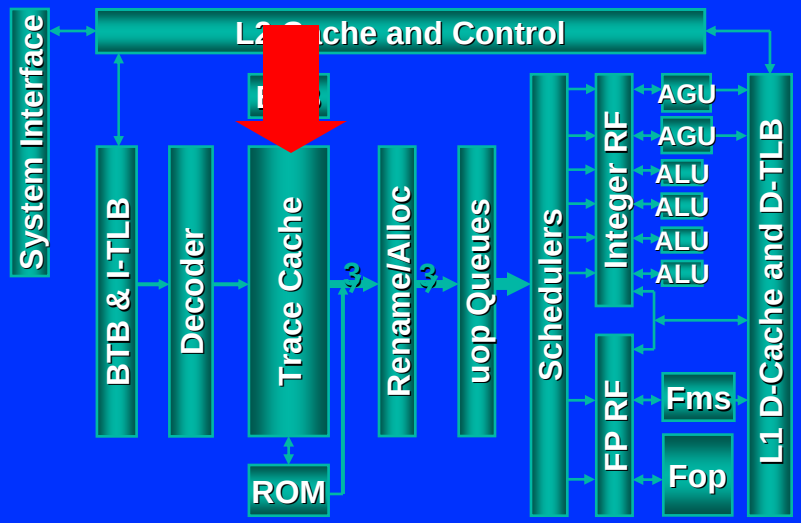
Intel
Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

TC Nxt IP: Trace cache next instruction pointer

Pointer from the BTB, indicating location of next instruction.



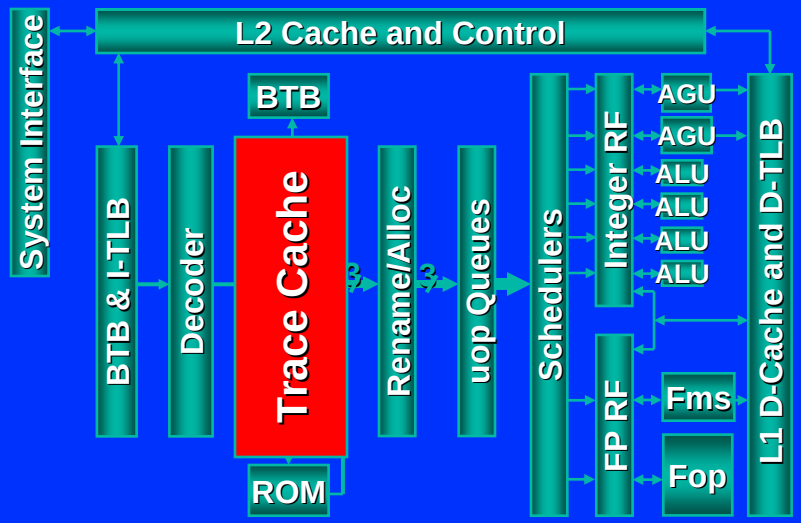
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

TC Fetch: Trace cache fetch

Read the decoded instructions (uOPs) out of the Execution Trace Cache



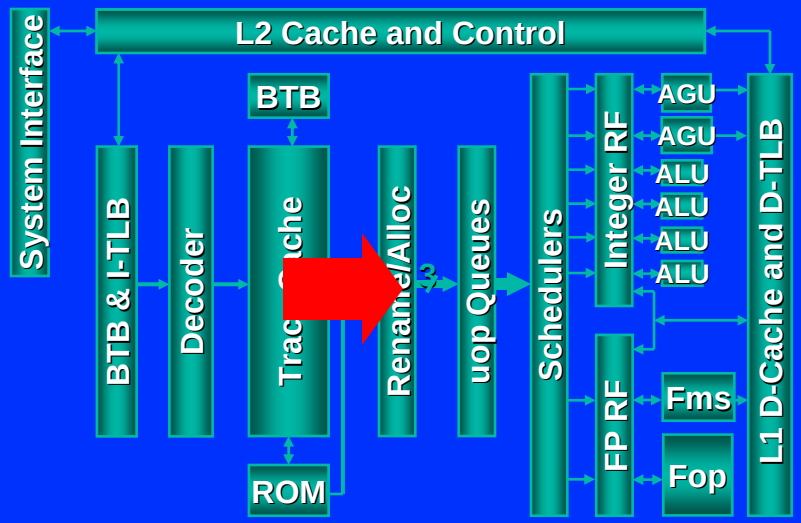
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Drive: Wire delay

Drive the uOPs to the allocator



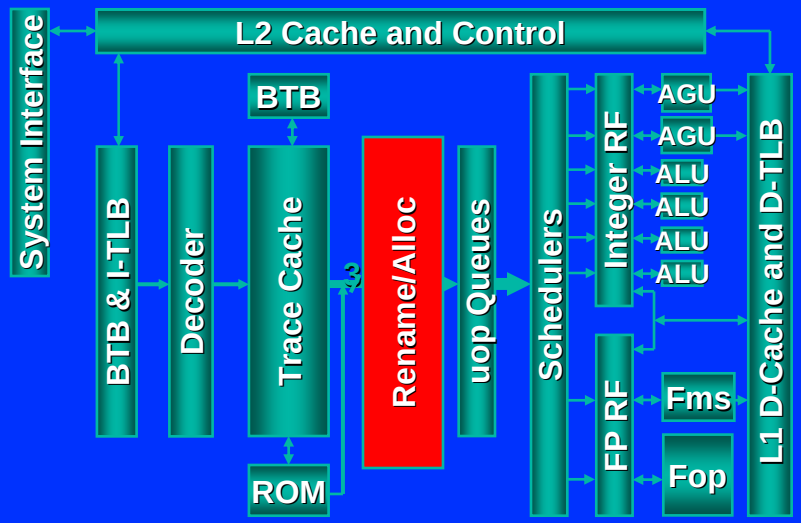
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Alloc: Allocate

Allocate resources required for execution. The resources include Load buffers, Store buffers, etc..



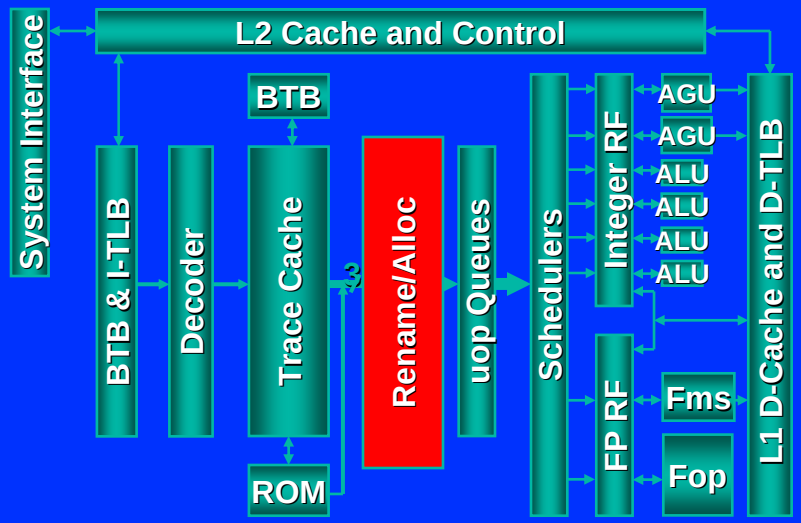
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Rename: Register renaming

Rename the logical registers (EAX) to the physical register space (128 are implemented).



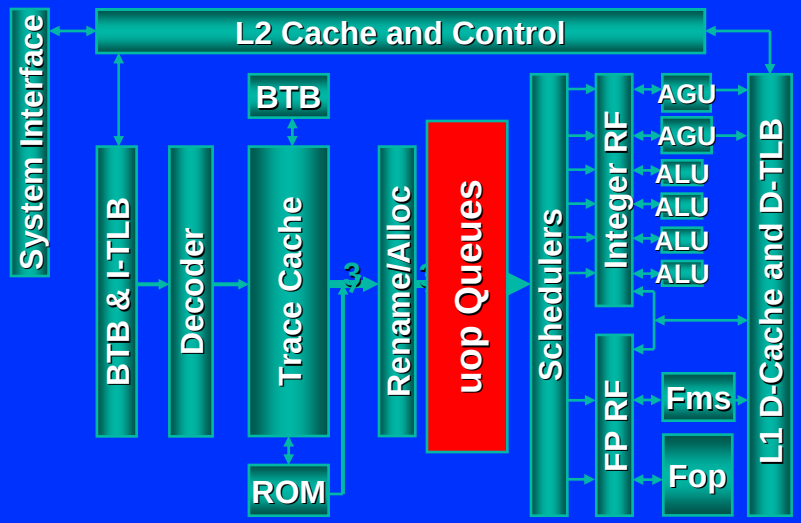
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Que: Write into the uOP Queue

uOPs are placed into the queues, where they are held until there is room in the schedulers



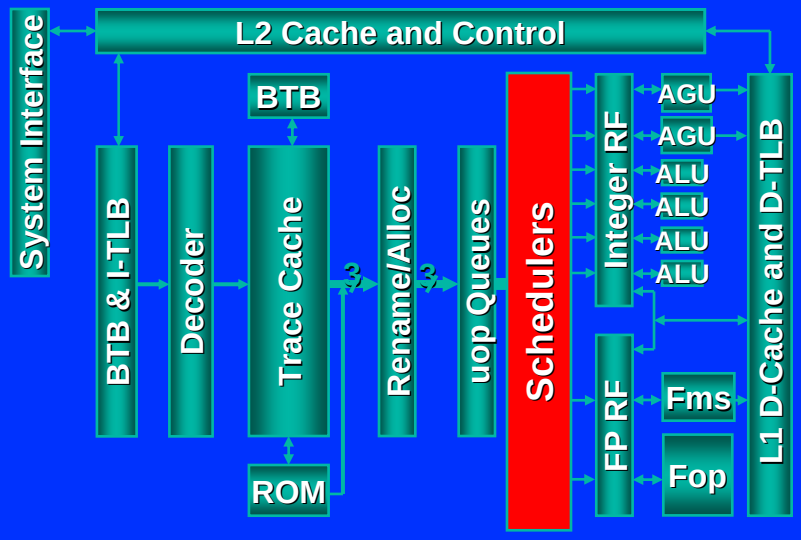
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Sch: Schedule

Write into the schedulers and compute dependencies. Watch for dependency to resolve.



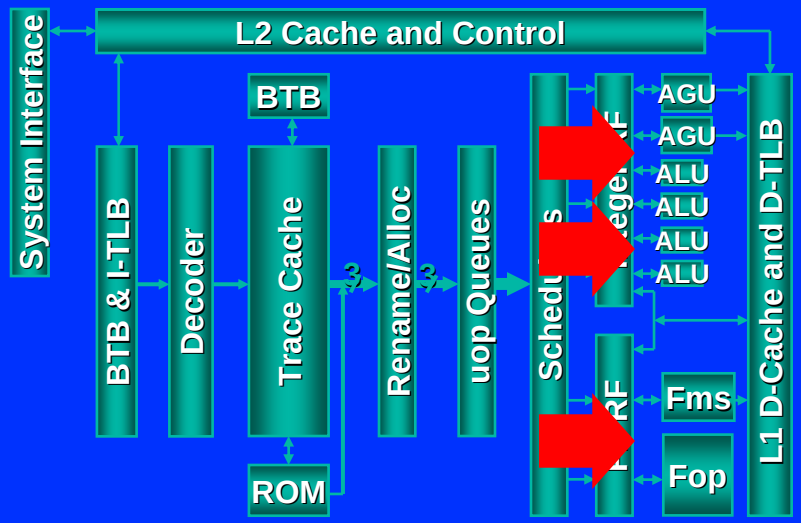
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Disp: Dispatch

Send the uOPs to the appropriate execution unit.



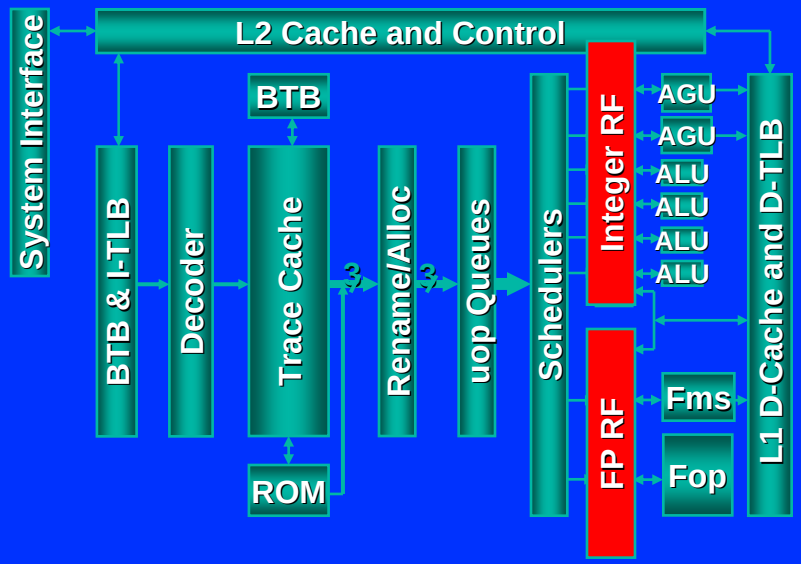
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

RF: Register File

Read the register file. These are the source(s) for the pending operation (ALU or other).



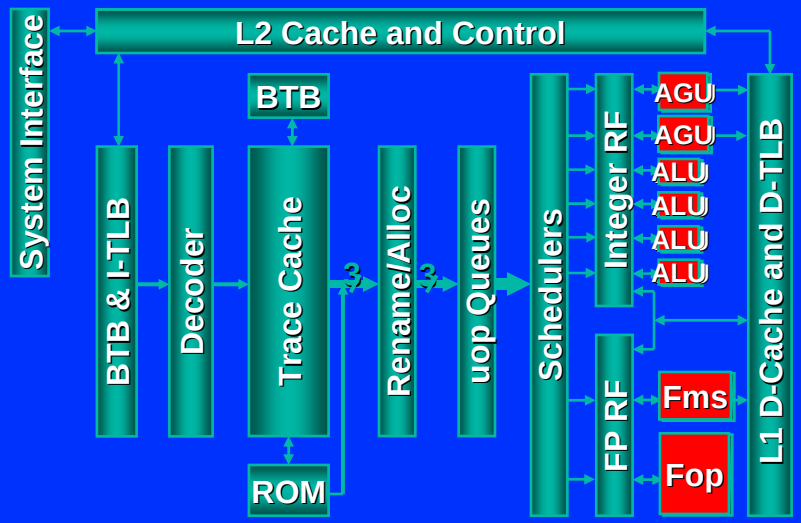
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Ex: Execute

Execute the uOPs on the appropriate execution port.



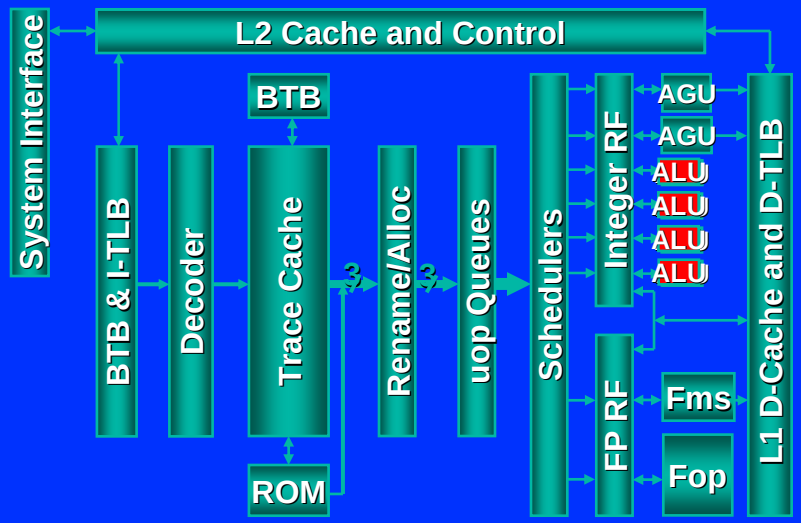
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Flgs: Flags

Compute flags (zero, negative, etc..). These are typically the input to a branch instruction.



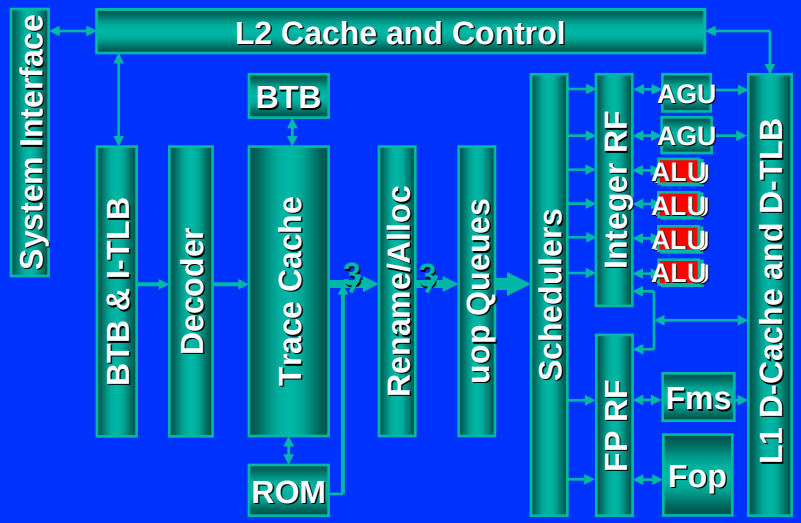
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

Br Ck: Branch Check

The branch operation compares the result of the actual branch direction with the prediction.



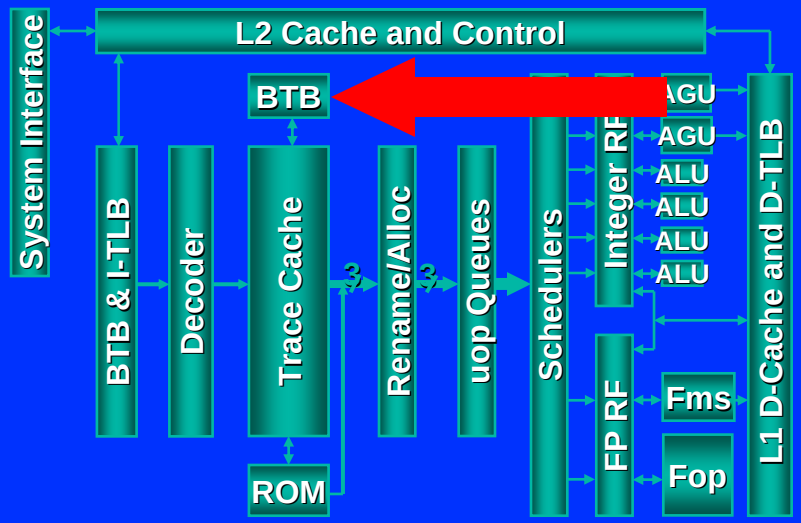
Intel Austin

Hyper Pipelined Technology

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TC Nxt IP		TC Fetch		Drive	Alloc	Rename		Que	Sch	Sch	Sch	Disp	Disp	RF	RF	Ex	Flgs	Br Ck	Drive

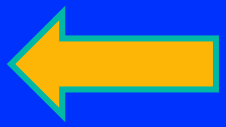
Drive: Wire delay

Drive the result of the branch check to the front end of the machine.



Intel Austin

Agenda

- IA-32 Processor Roadmap
- Design Goals
- Architecture 101
  - Frequency
  - Instructions Per Cycle 
- Advanced Architectural Concepts
  - Data Speculation
  - Data Prefetching
  - HyperThreading™
- Summary

Intel
Austin

CPU Architecture 101

Intel
Austin

Improving IPC

- Improved Branch Predictor
- Trace Cache
- Large Instruction Window
- Reduce operation latencies
  - Faster ALUs
  - Faster L1 data cache
  - Reduce average memory latency
- Increase bandwidth

Intel Austin

Branch Prediction

- Accurate branch prediction is key to tolerating longer pipelines
- Dramatic improvement over P6 branch predictor:
  - 8x the size (4K)
  - Eliminated 1/3 of the mispredictions
- Higher accuracy than *all* other publicly disclosed algorithms
  - (g-share, hybrid, etc)

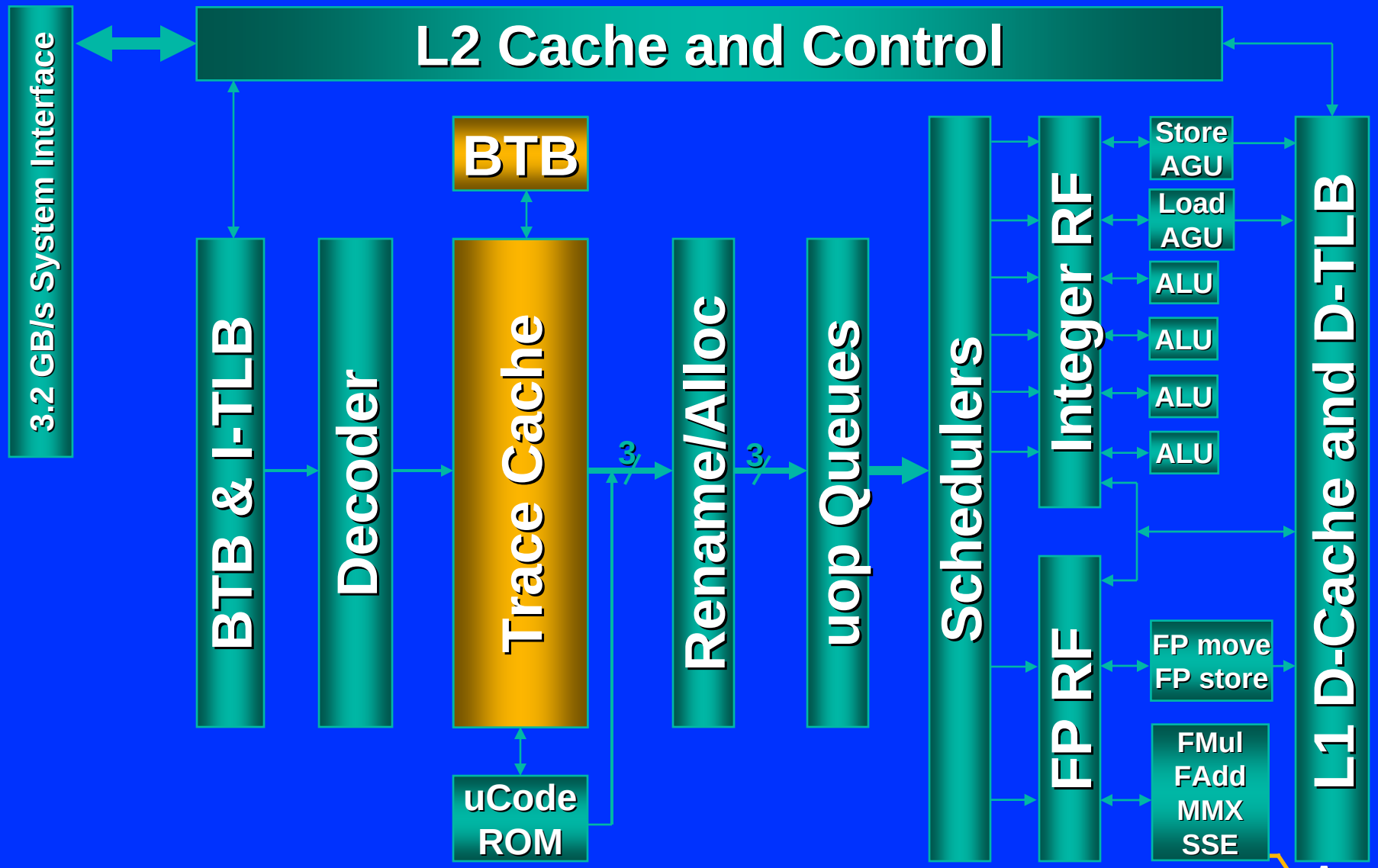
Intel Austin

Improving IPC

- Improved Branch Predictor
- Trace Cache
- Large Instruction Window
- Reduce operation latencies
  - Faster ALUs
  - Faster L1 data cache
  - Reduce average memory latency
- Increase bandwidth

Intel Austin

Trace Cache



Trace Cache

- Lowest level instruction cache
  - Caches “decoded” IA-32 instructions (uops)
  - Capacity ~12K uOps
- Addresses difficulty of parallel CISC decode
- Removes decoder pipeline latency from branch miss loop
- Integrates branches into single line
  - Removes branch fetch bubbles

Intel Austin

Trace Cache: Code Example

```
T0:     1  cmp
        2  br -> T1
        ..
        ...  (unused code)

T1:     3  sub
        4  br -> T2
        ..
        ...  (unused code)

T2:     5  mov
        6  sub
        7  br -> T3
        ..
        ...  (unused code)

T3:     8  add
        9  sub
        10 mul
        11 cmp
        12 br  -> T0
```

Trace Cache Delivery

1 T0:cmp	2 br T1	3 T1:sub
4 br T2	5 mov	6 sub
7 br T3	8 T3:add	9 sub
10 mul	11 cmp	12 br T0

Intel Austin

Improving IPC

- Improved Branch Predictor
- Trace Cache
- **Large Instruction Window**
- Reduce operation latencies
  - Faster ALUs
  - Faster L1 data cache
  - Reduce average memory latency
- Increase bandwidth

Intel Austin

Large Instruction Window

- Search window for independent uops
- 126 uops in flight (3x P6)
- 48 loads (3x P6) and 24 stores (2x P6)

Deep Speculation Improves IPC

Intel Austin

Improving IPC

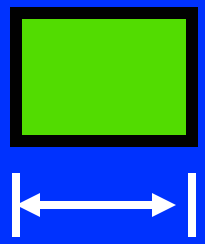
- Improved Branch Predictor
- Trace Cache
- Large Instruction Window
- Reduce operation latencies
  - Faster ALUs
  - Faster L1 data cache
  - Reduce average memory latency
- Increase bandwidth

Intel Austin

Lower ALU latency

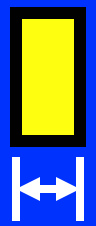
In the same process:

- P6:
  - 1 clock @ 1GHz



1ns

- P4P:
  - ½ clock @ >1.4GHz



<0.36ns

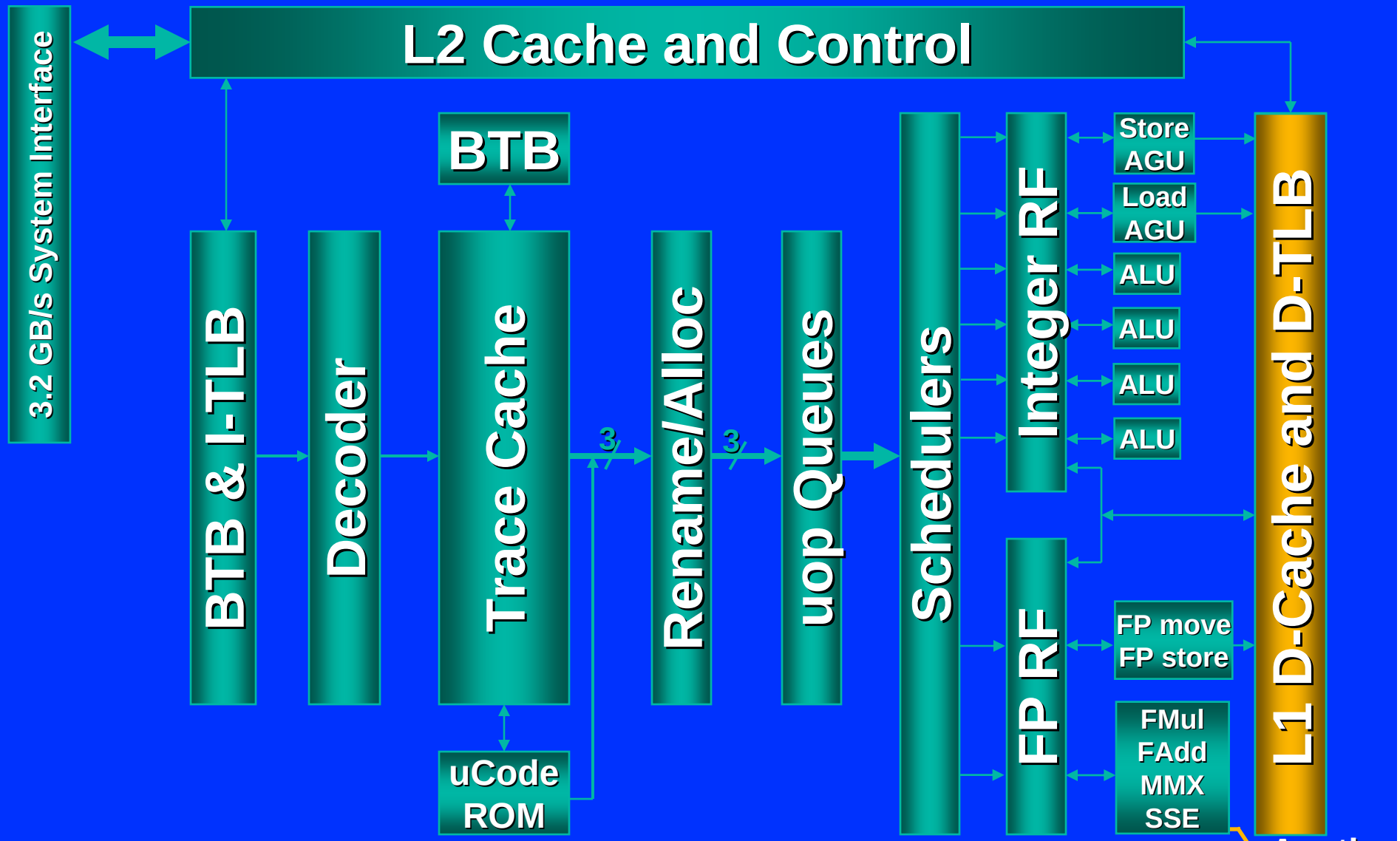
Intel Austin

Improving IPC

- Improved Branch Predictor
- Trace Cache
- Large Instruction Window
- Reduce operation latencies
  - Faster ALUs
  - Faster L1 data cache
  - Reduce average memory latency
- Increase bandwidth

Intel Austin

L1 Data Cache



L1 Data Cache

- 8KB, 4-way set associative, 64-byte lines
- Very high bandwidth
  - 1 Ld and 1 St / clock (vs. 1 Ld *or* 1 St in P6)
- New access algorithms reduce latency
  - 2 cycle loads (vs 3 cycles in P6)

New algorithms/circuits enable faster cache

Intel Austin

L1 Access: Data Speculation

- Observation: Almost all memory accesses hit in the cache
- Optimize for the common case
  - Assume that the access will hit the cache
  - Use an efficient mechanism to fix the rare cases that miss
- Benefit:
  - Reduces latency
  - Significantly higher performance

Intel Austin

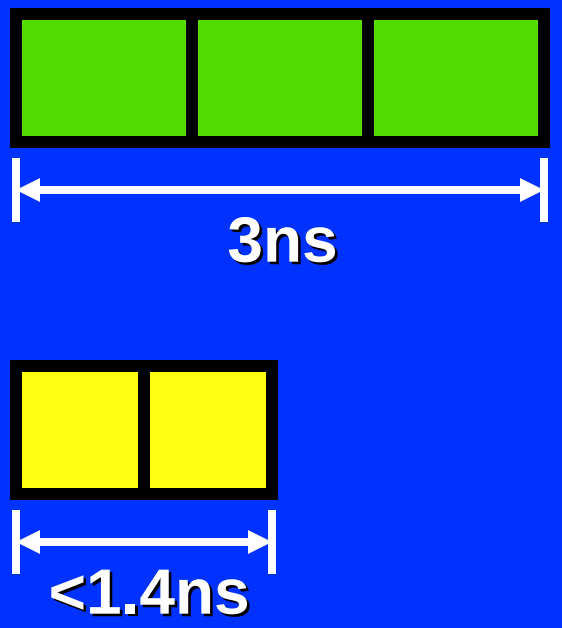
L1 Access: Replay

- Repairs incorrect speculation
  - Re-execute until correct
- Replay is uOP specific
  - Re-execute the uOP that mis-speculated
  - Re-execute only dependent uOPs
  - Independent uOPs are not re-executed

Intel Austin

L1 Cache is >2x Faster

- P6:
  - 3 clocks @ 1GHz
- P4P:
  - 2 clocks @ ≥1.4GHz



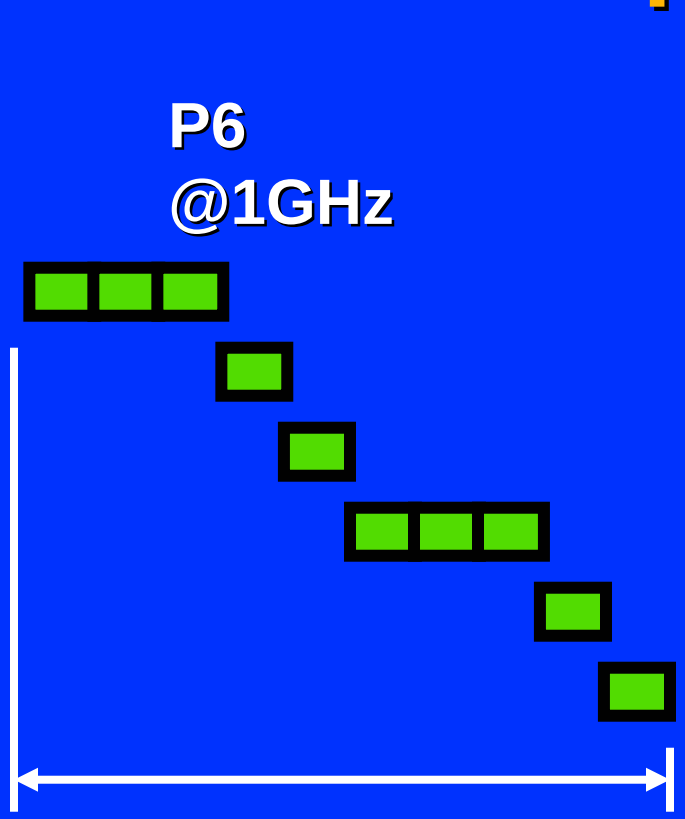
Lower Latency is Higher Performance

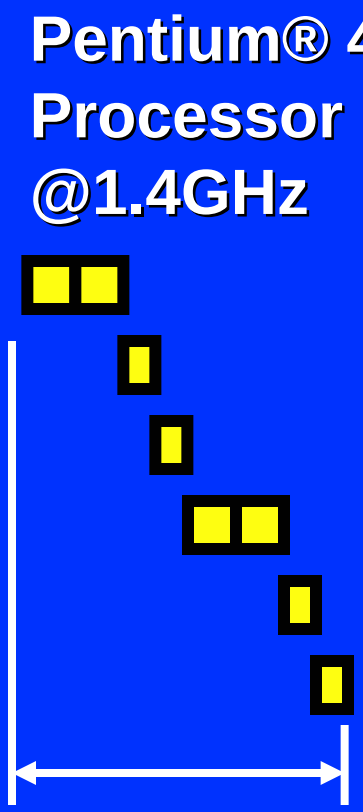
Intel Austin

L1 Cache: Code Example

Code Sequence

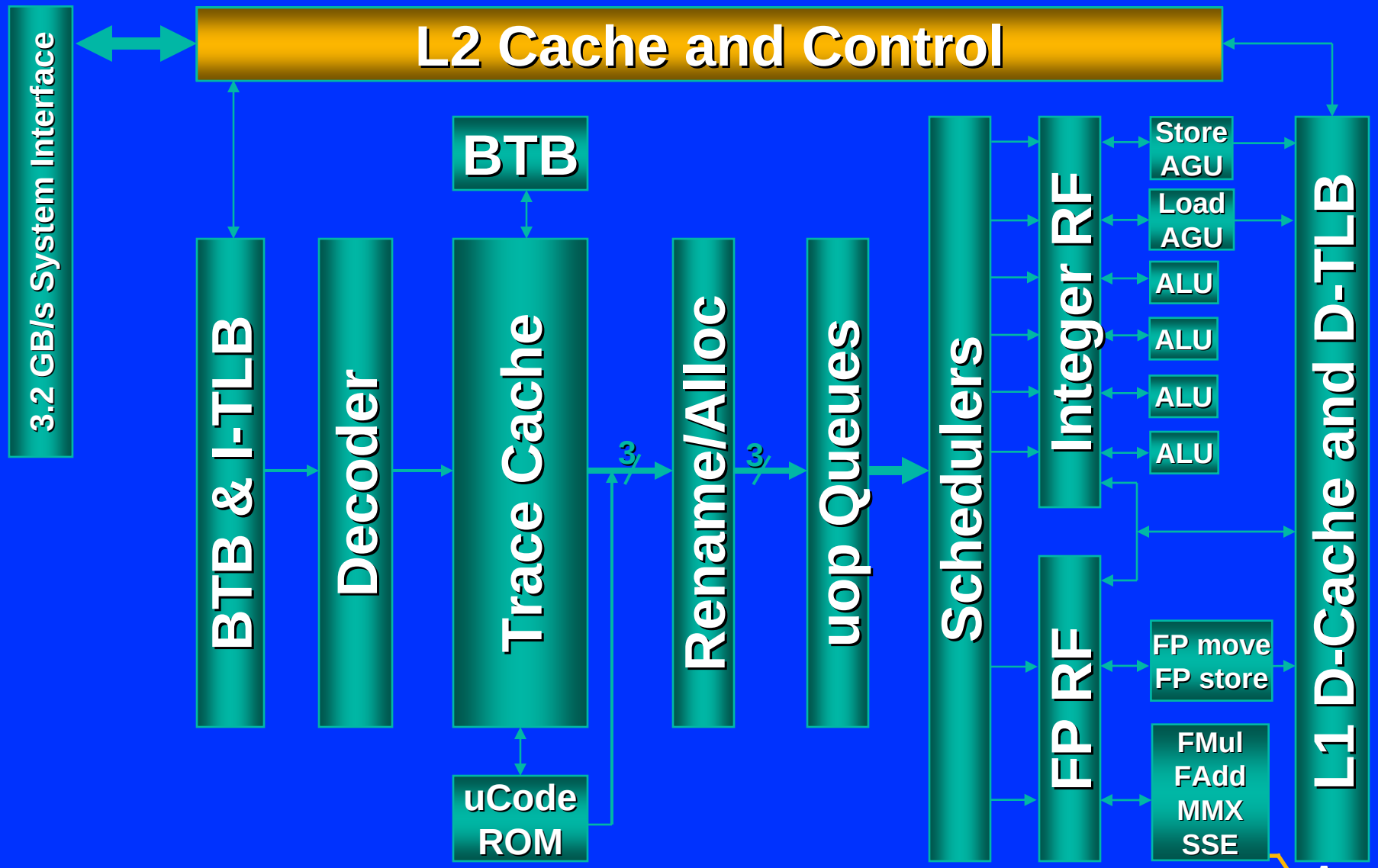
```
Ld
Add
Add
Ld
Add
Add
```





Intel Austin

L2 Cache



Unified L2 Cache

- 256KB, 8-way set associative
  - 128-byte lines
  - Two sectors per line
- 45 GB/Sec bandwidth @ 1.4GHz
  - 2.8x P6 @1GHz

Intel Austin

Improving IPC

- Improved Branch Predictor
- Trace Cache
- Large Instruction Window
- Reduce operation latencies
  - Faster ALUs
  - Faster L1 data cache
  - Reduce average memory latency
- Increase bandwidth

Intel Austin

Aggregate Cache Latency

- Function of all caches in a processor
- Overall Effective Latency

  L1 latency +

  L1 Miss Rate * L2 latency +

  L2 Miss Rate * DRAM Latency

Average load latency is >1.8x lower than the Pentium® III Processor

Average on desktop applications,
Pentium® III Processor @ 1GHz, Pentium® 4 Processor @ 1.4GHz

Intel Austin

Improving IPC

- Improved Branch Predictor
- Trace Cache
- Large Instruction Window
- Reduce operation latencies
  - Faster ALUs
  - Faster L1 data cache
  - Reduce average memory latency
- Increase bandwidth

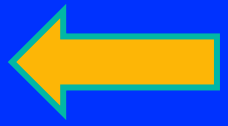
Intel Austin

Higher BW & Lower Latency

	Pentium®III Processor	Pentium® 4 Processor	Relative Improvement
Frequency	1 GHz	> 1.4 Ghz	> 1.4
Adder Speed	1 ns	< .36 ns	> 2.8
L1 Cache Speed	3 ns	< 1.42 ns	> 2.1
L1 Cache Size	16 KB	8 KB	0.5
L1 Cache Bandwidth	16 GB/sec	> 44.8 GB/sec	> 2.8
L2 Cache Bandwidth	16 GB/sec	> 44.8 GB/sec	> 2.8
Uop Fetch Bandwidth	3 billion/sec	> 4.2 billion/sec	> 1.4
Adder Bandwidth	2 billion/sec	> 5.6 billion/sec	> 2.8
Branch targets	512	4096	8
Instructions In flight	40	126	3.15
Loads in flight	16	48	3
Stores in flight	12	24	2

Intel Austin

Agenda

- IA-32 Processor Roadmap
- Design Goals
- Architecture 101
  - Frequency
  - Instructions Per Cycle
- Advanced Architectural Concepts
  - Data Speculation 
  - Data Prefetching
  - HyperThreading™
- Summary

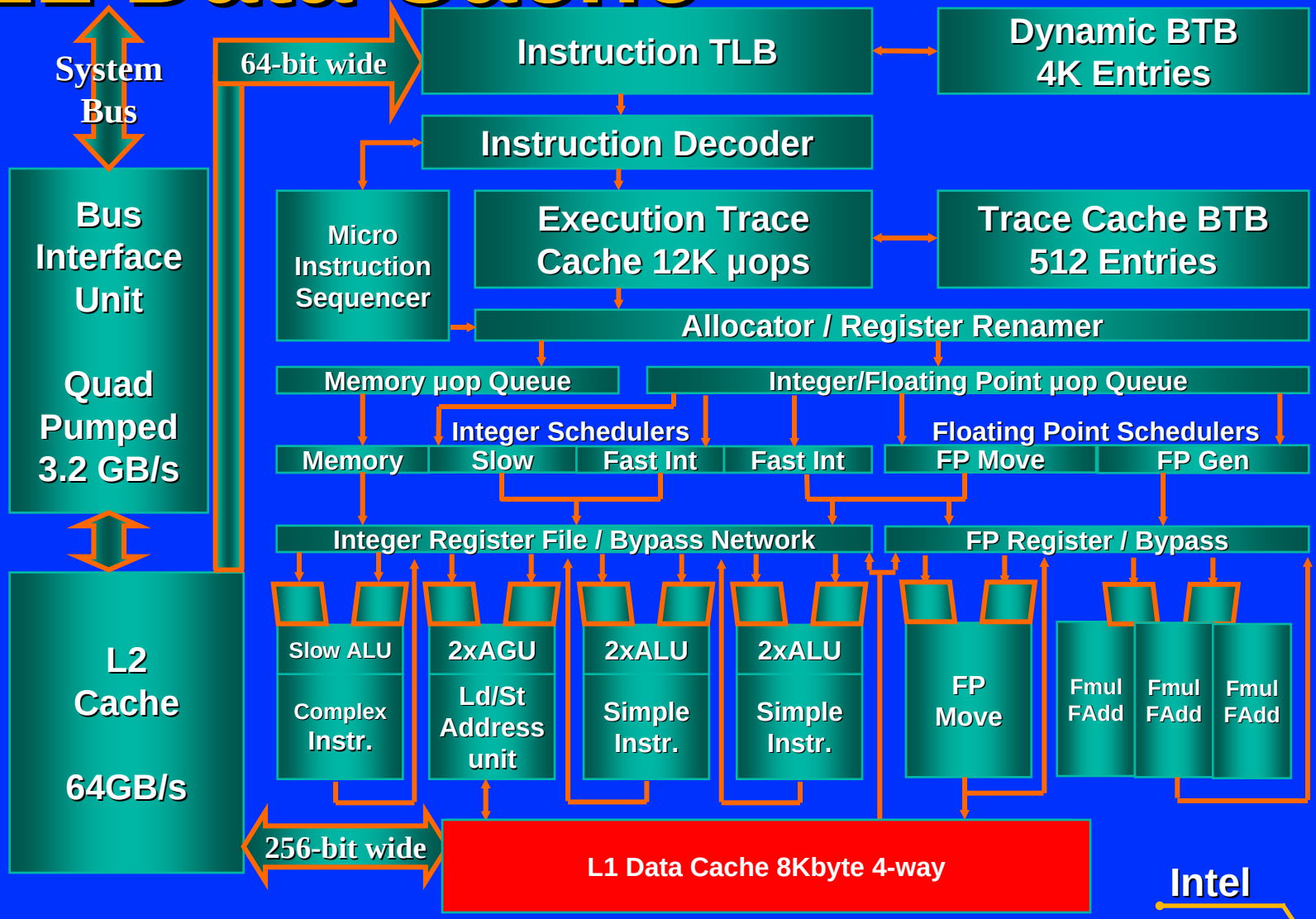
Intel Austin

Data Speculation

- Use data before we are sure it is valid
  - Lowers effective LD latency
  - Fast ALUs in Pentium 4 want fast LDs
  - Ratio of LD latency to ADD latency is important if 1 in 5 uops is a LD
- As pipelines get deeper, data speculation gets more important
  - Number of cycles saved /w data speculation increases as pipeline depth increases

Intel Austin

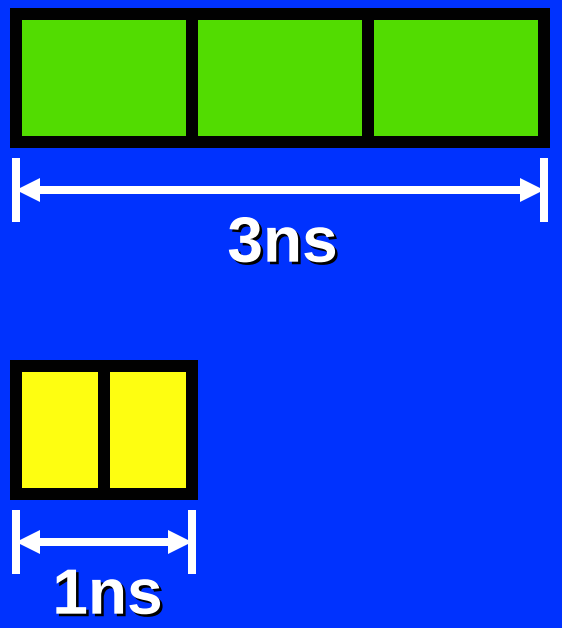
L1 Data Cache



Intel Austin

L1 Cache is >3x Faster

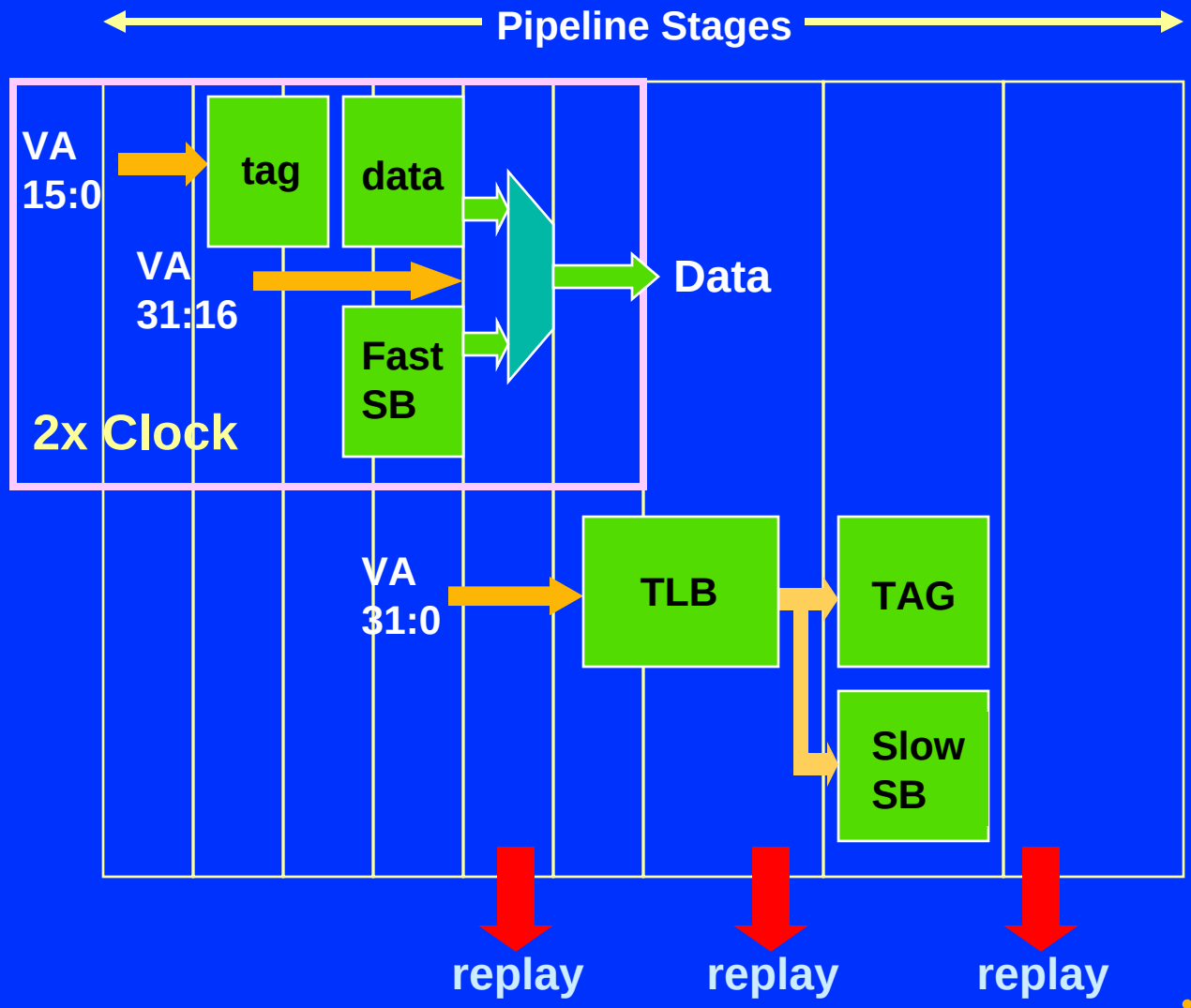
- P6:
  - 3 clocks @ 1GHz
- P4P:
  - 2 clocks @ 2GHz



Lower Latency is Higher Performance

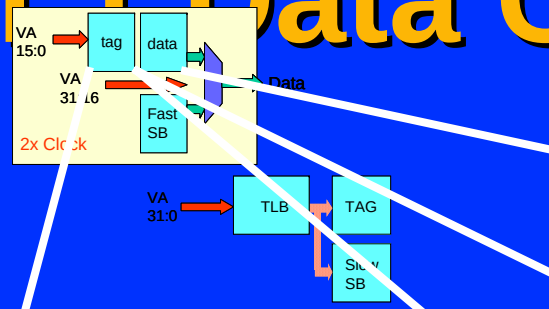
Intel Austin

L1 Data Cache



Intel
Austin

L 1 Data Cache



Way Predictor (Tag array)

Data array

Way
select

Hit
(Replay)

10:6

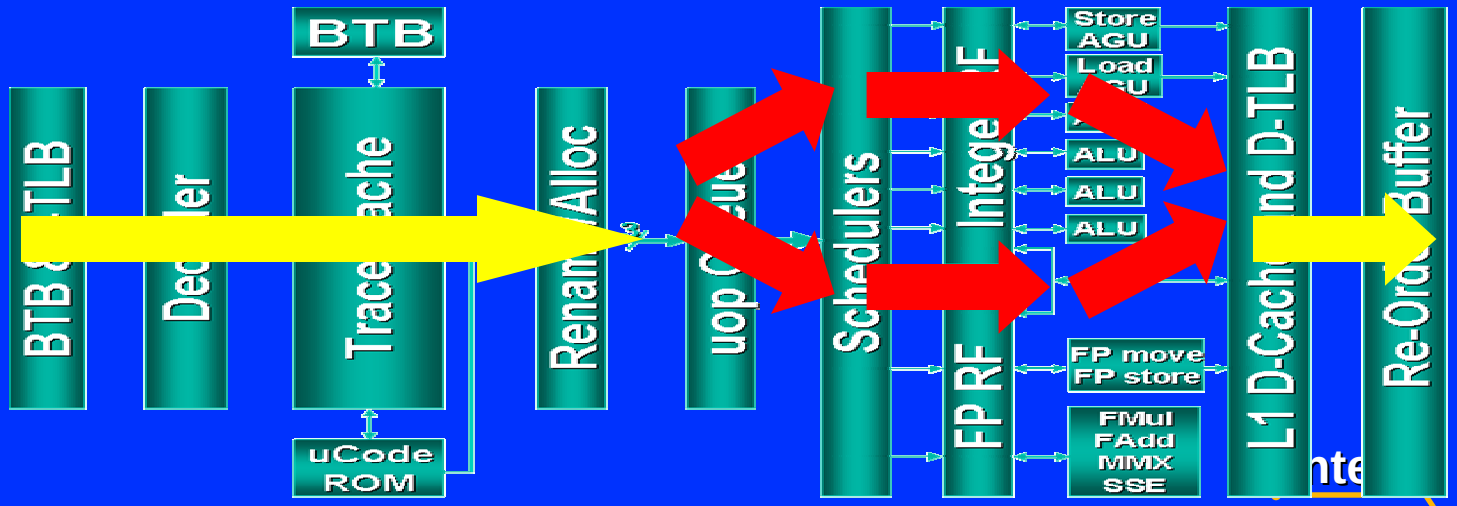
15:11

19:16

Intel
Austin

A Digression on Stores

- Two components to a store:
  - STA: address computation
  - STD: data piece
- Hybrid uOP
  - Single uOP in the front, back ends
  - Two uOPs in the middle



Memory Disambiguation

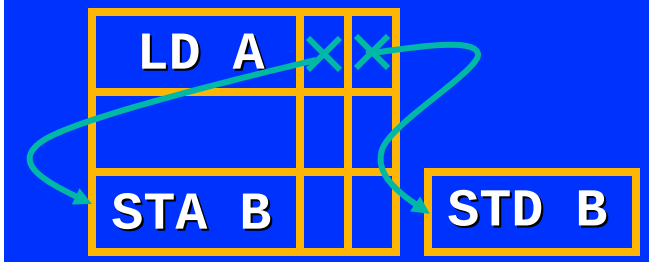
```
Ld EAX <- AddrA
```

- If the store is older
  - And AddrA = AddrB
  - Then the load must get DataB
- Dependencies can not be resolved until execution

Intel Austin

Memory Disambiguation

Option 1



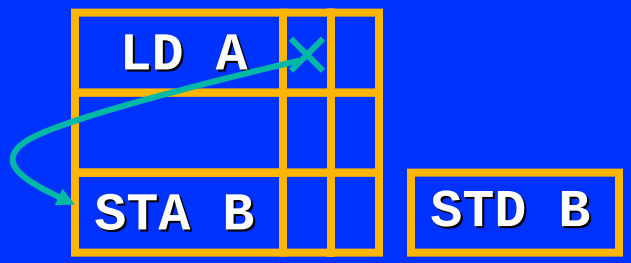
Loads wait for:

- All older STAs AND
- All older STDs

No Recovery

K7

Option 2



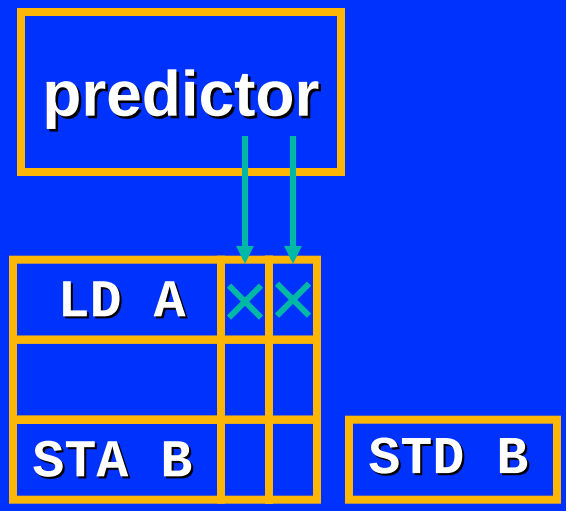
Loads wait for:

- All older STAs

STD Recovery

P4

Option 3



Predict

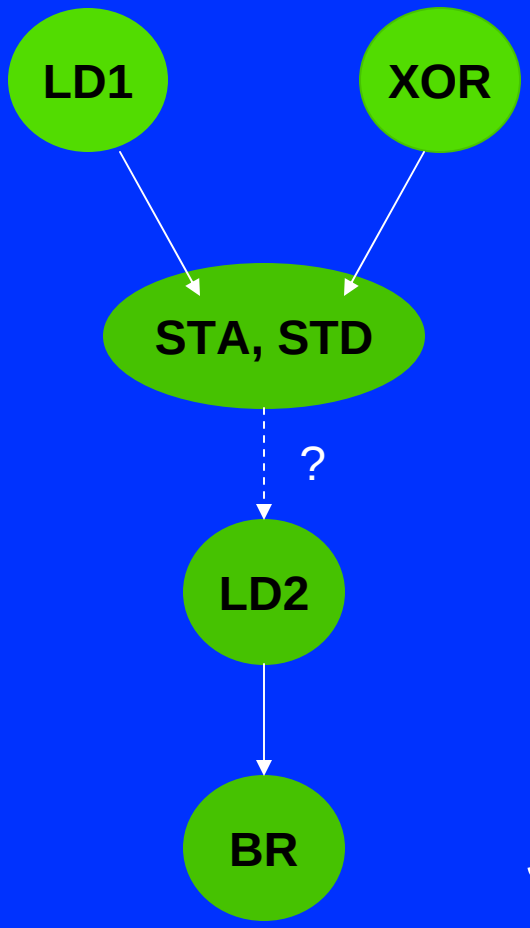
- Specific older STAs
- Specific older STDs

Complex Recovery

EV8

Intel Austin

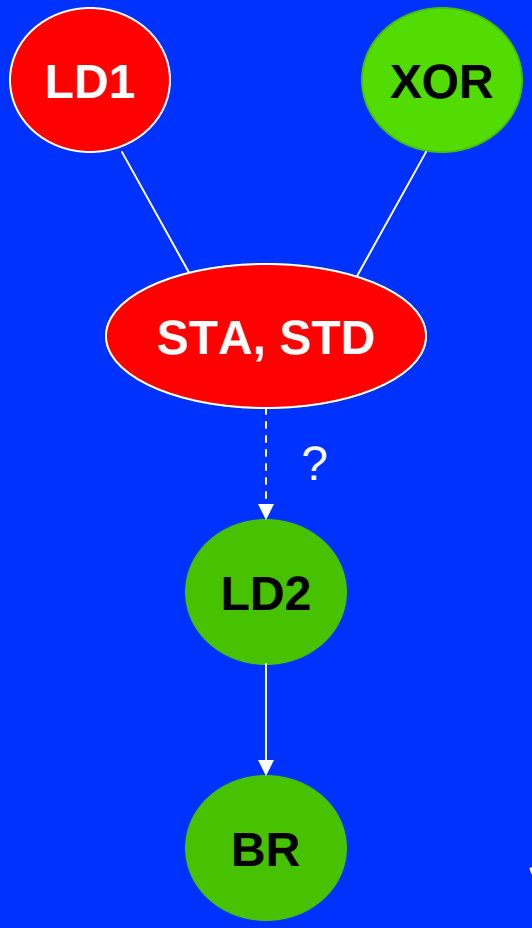
Example



- ST writes new value
- LD2 forwards
- Branch resolves based on new value

Intel Austin

Example



ML=0

LD misses, replays

ML=1

STA dependent, replays

LD hits, gets old value

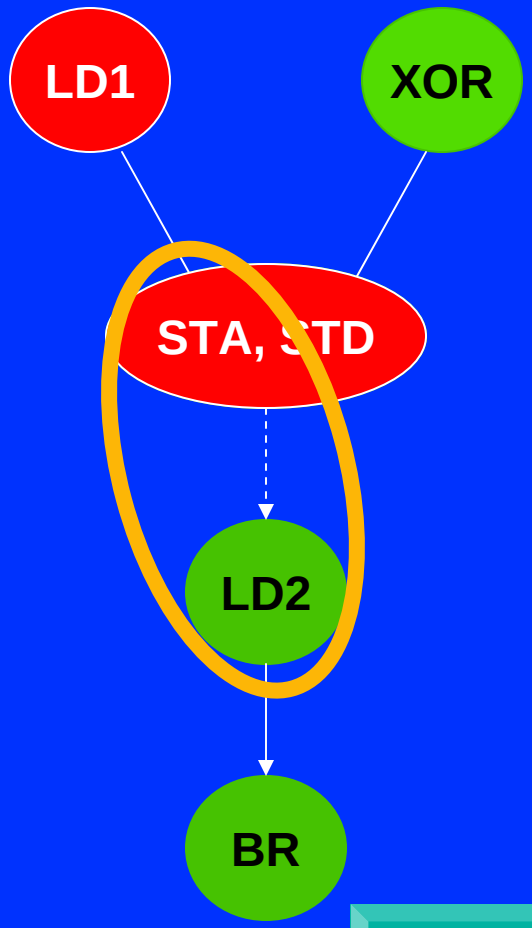
BR mispredicts

JMP if ML=0

Bogus JeClear

Intel Austin

Bogus JeClear

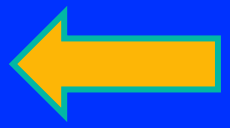


- If LD2 depends on the STA, they are usually part of the same dataflow graph
- If the STA replays, the LD usually has an address dependence that will prevent bogus JeClears

But, there are exceptions…

Intel Austin

Agenda

- IA-32 Processor Roadmap
- Design Goals
- Architecture 101
  - Frequency
  - Instructions Per Cycle
- Advanced Architectural Concepts
  - Data Speculation
  - Data Prefetching 
  - HyperThreading™
- Summary

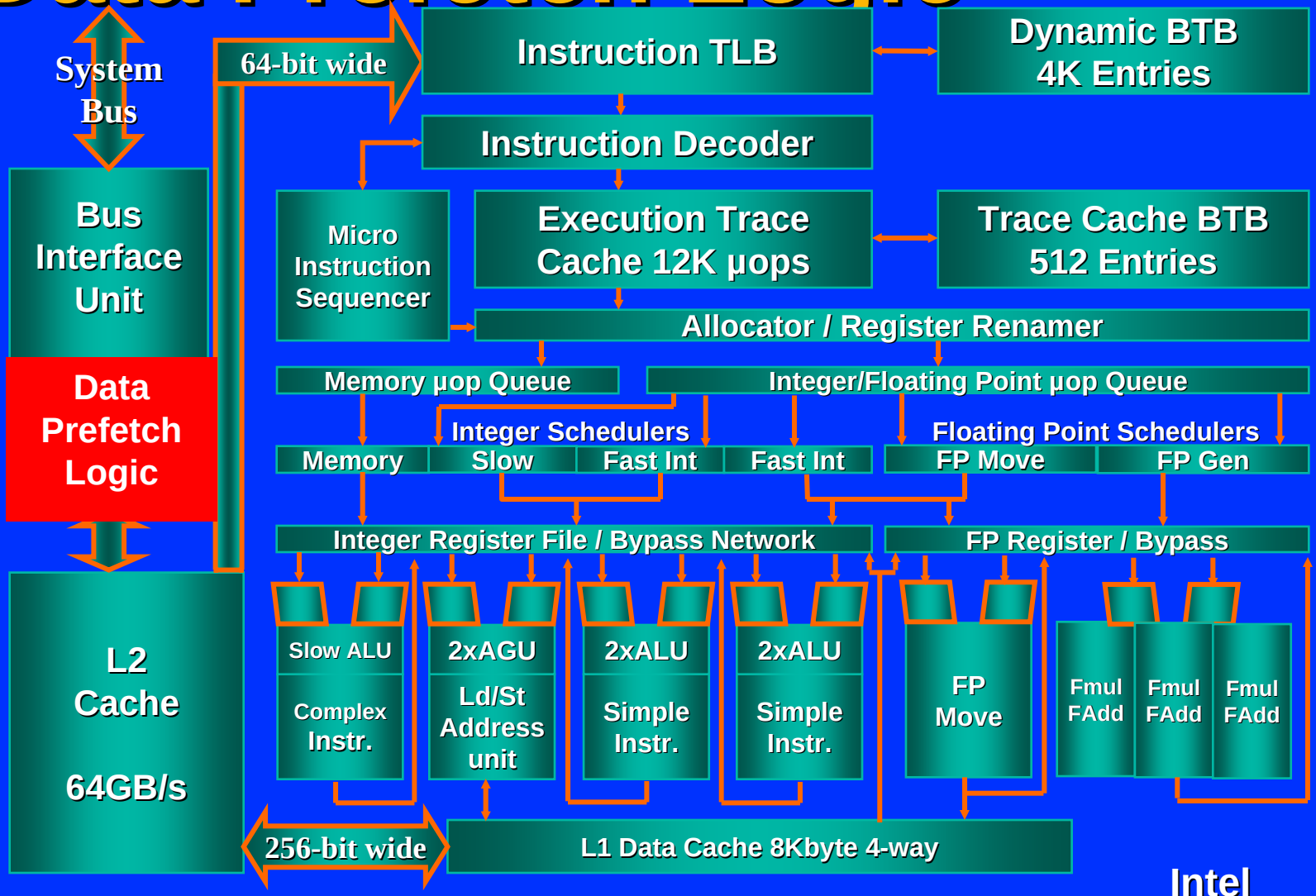
Intel Austin

Reducing Latency

- As frequency increases, it is important to improve the performance of the memory subsystem
- Data Prefetch Logic
  - Watches processor memory traffic
  - Looks for patterns
  - Initiates accesses

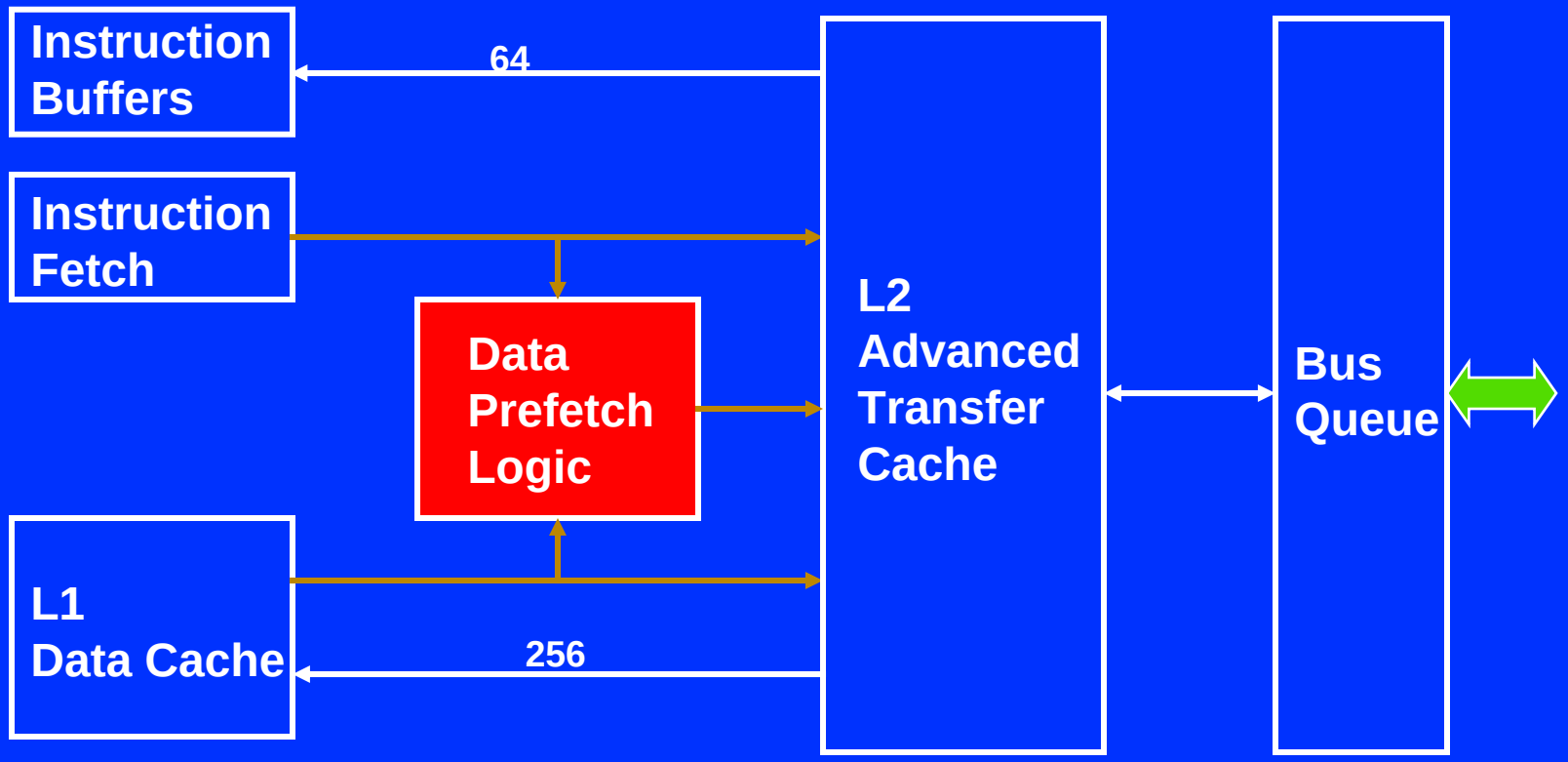
Intel Austin

Data Prefetch Logic



Intel Austin

Data Prefetch Logic



Prefetch logic first checks L2 cache and then fetches lines from memory that miss L2 cache.

Intel Austin

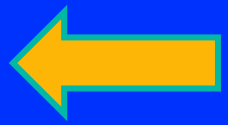
Data Prefetch Logic

- Watches for streaming memory access patterns
  - Can track 8 multiple independent streams
  - Loads, Stores or Instruction
  - Forward or Backward
- Analysis on 32 byte cache line granularity
- Looks for “mostly” complete streams:
  - Access to cache lines 1,2,3,4,5,6 will prefetch
  - Access to cache lines 1,2, 4,5,6 will prefetch
  - 1, ,3, , ,6, , ,9 will not prefetch

Intel Austin

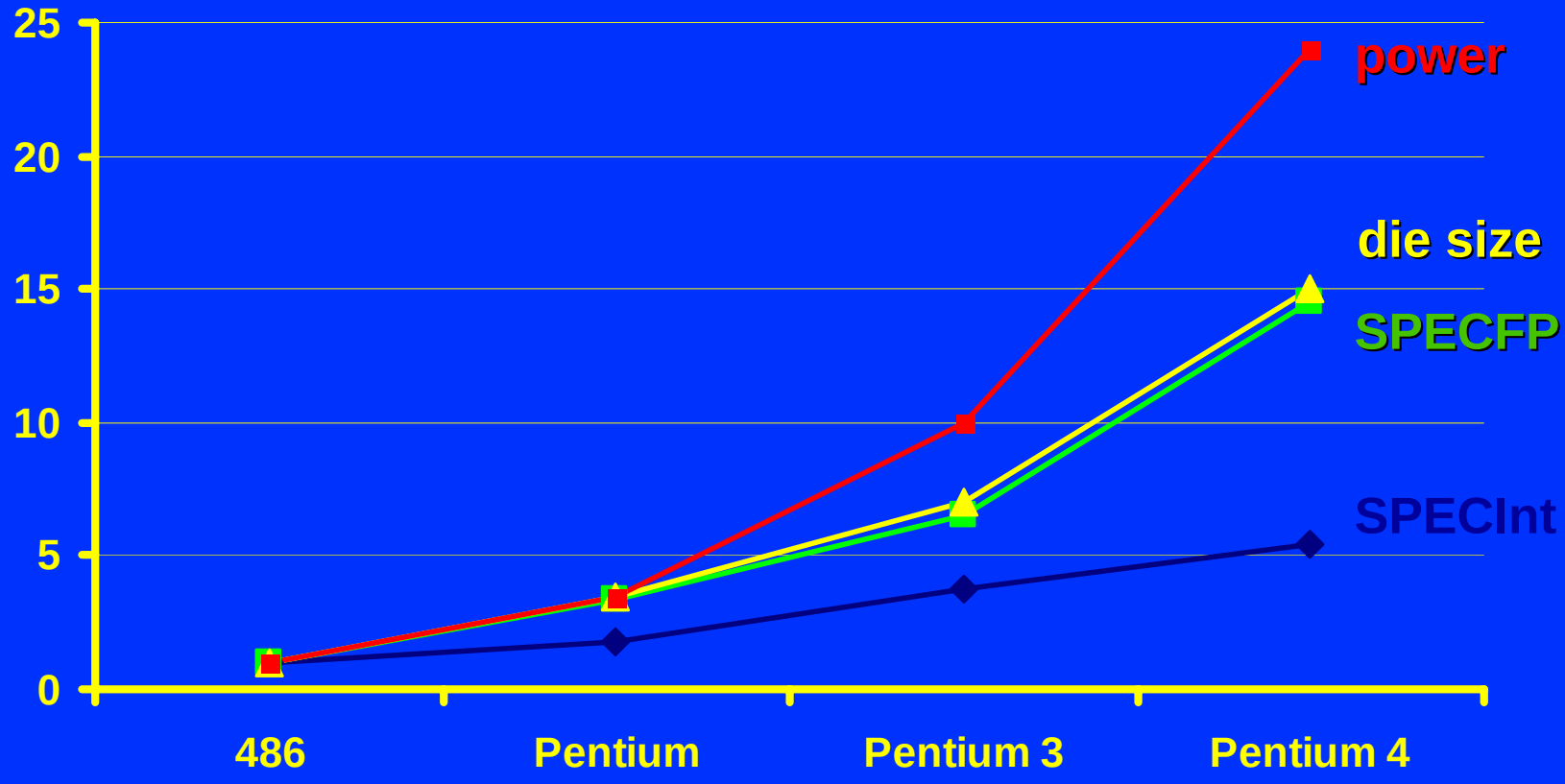
Agenda

- IA-32 Processor Roadmap
- Design Goals
- Architecture 101
  - Frequency
  - Instructions Per Cycle
- Advanced Architectural Concepts
  - Data Speculation
  - Data Prefetching
  - HyperThreading™
- Summary



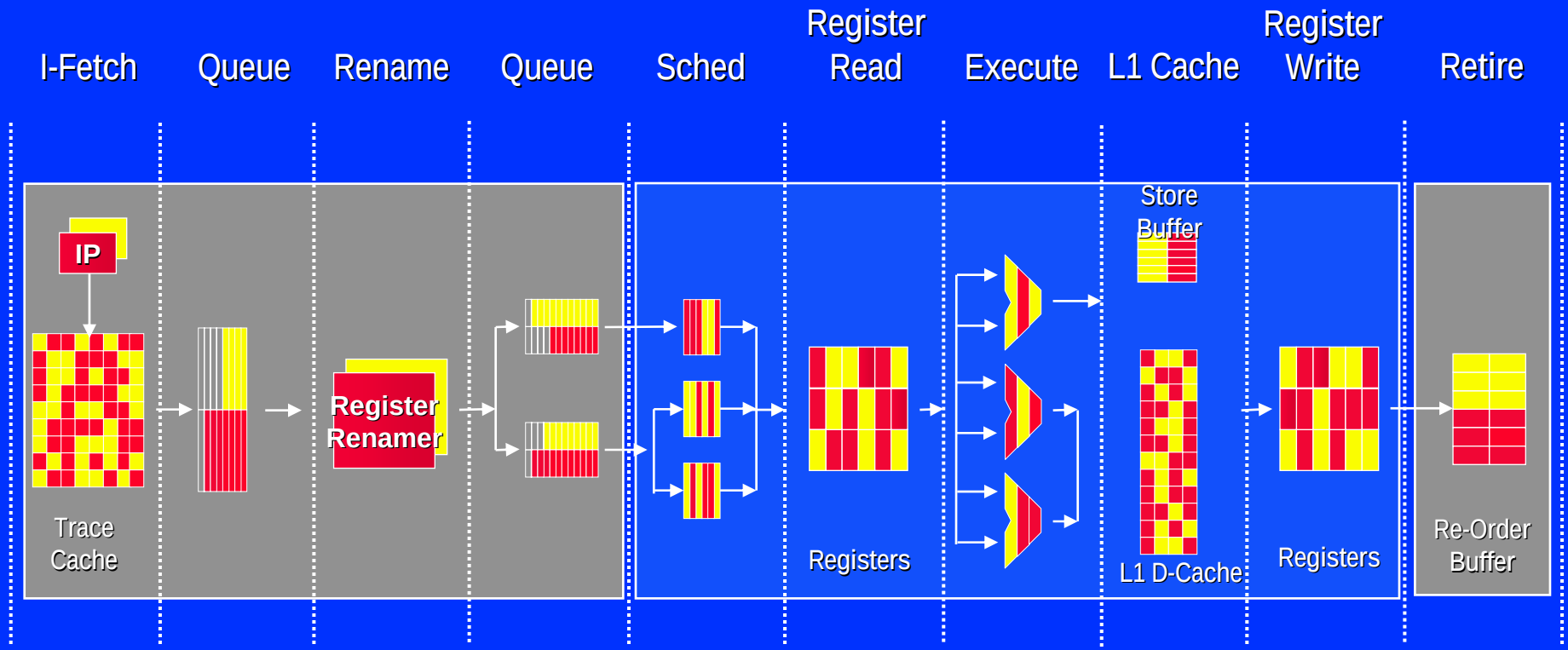
Intel Austin

Single-stream Performance vs Costs



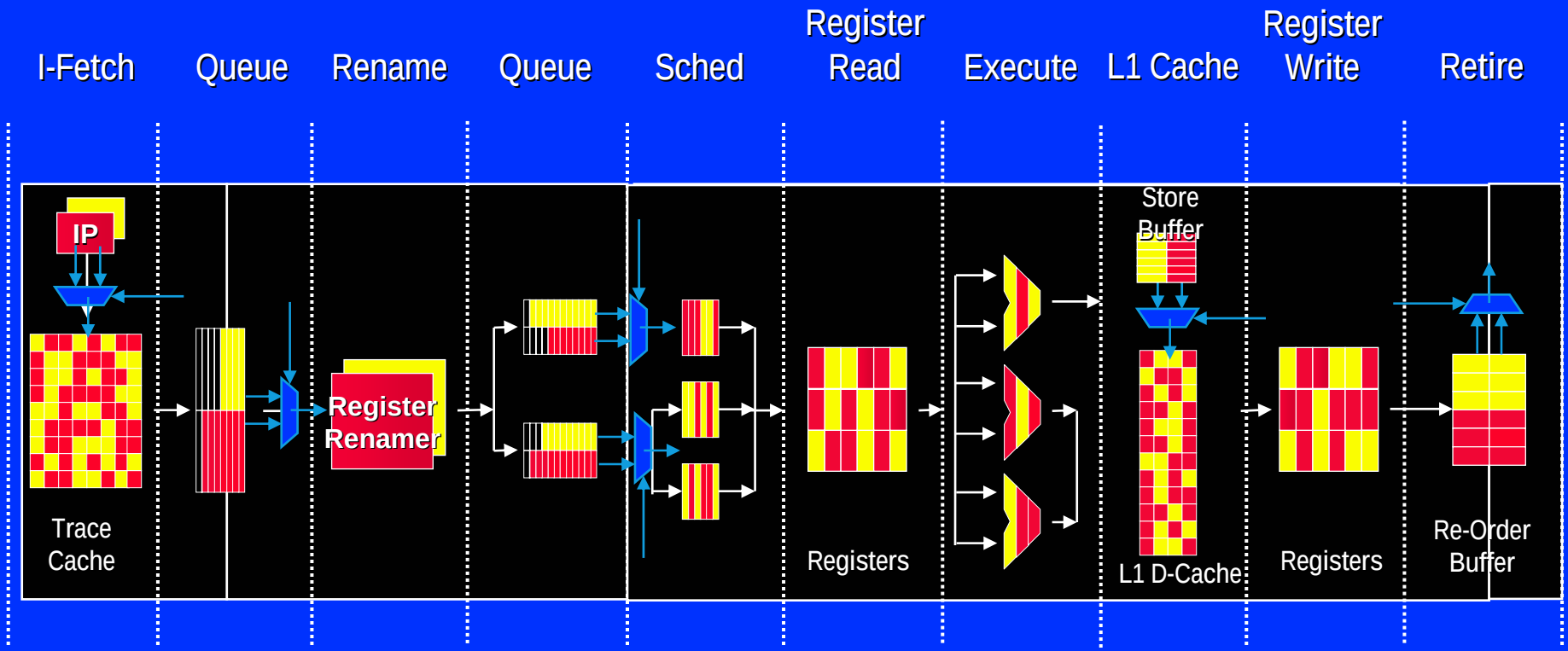
Intel
Austin

Basic Hyper-Threading Pipeline



Intel Austin

Thread-Selection Points



Intel Austin

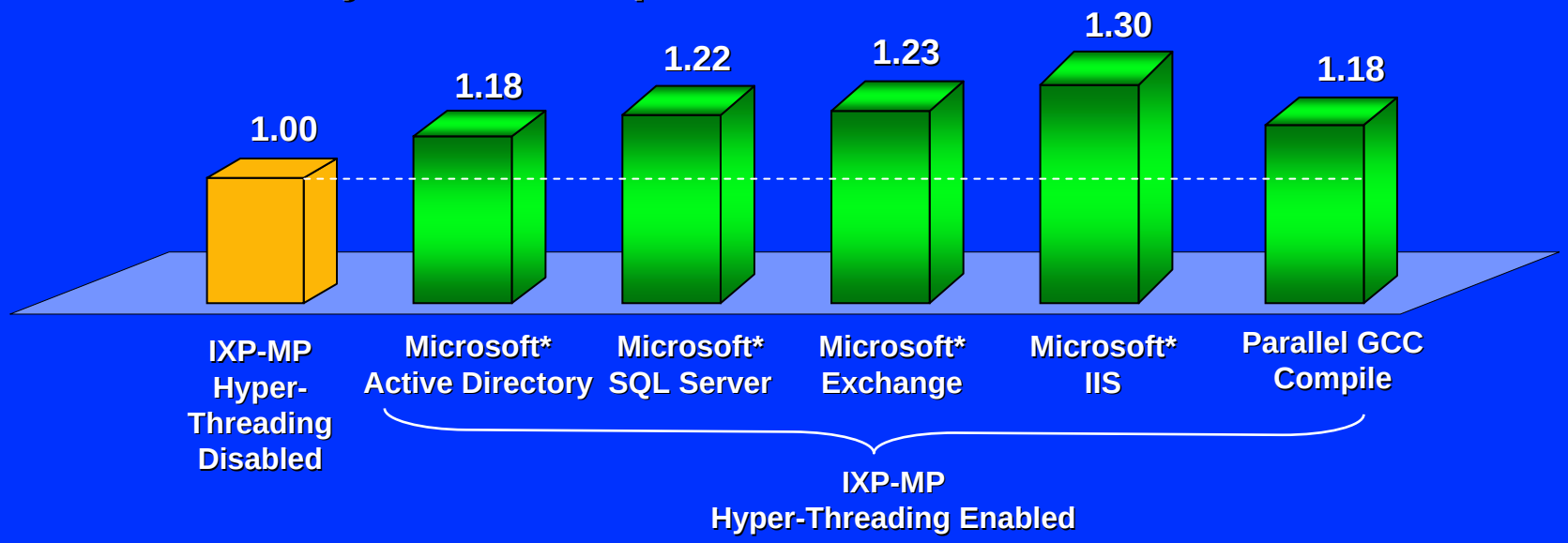
Performance Results

- Preliminary performance on a few applications
  - All applications unmodified
- Continuing evaluation over larger pool of applications
- Performance varies as expected with:
  - Number of parallel threads
  - Resource utilization

Intel Austin

Intel® Xeon™ Processor MP
Hyper-Threading Technology Performance

Application Measurements
by Intel Microprocessor Software Labs



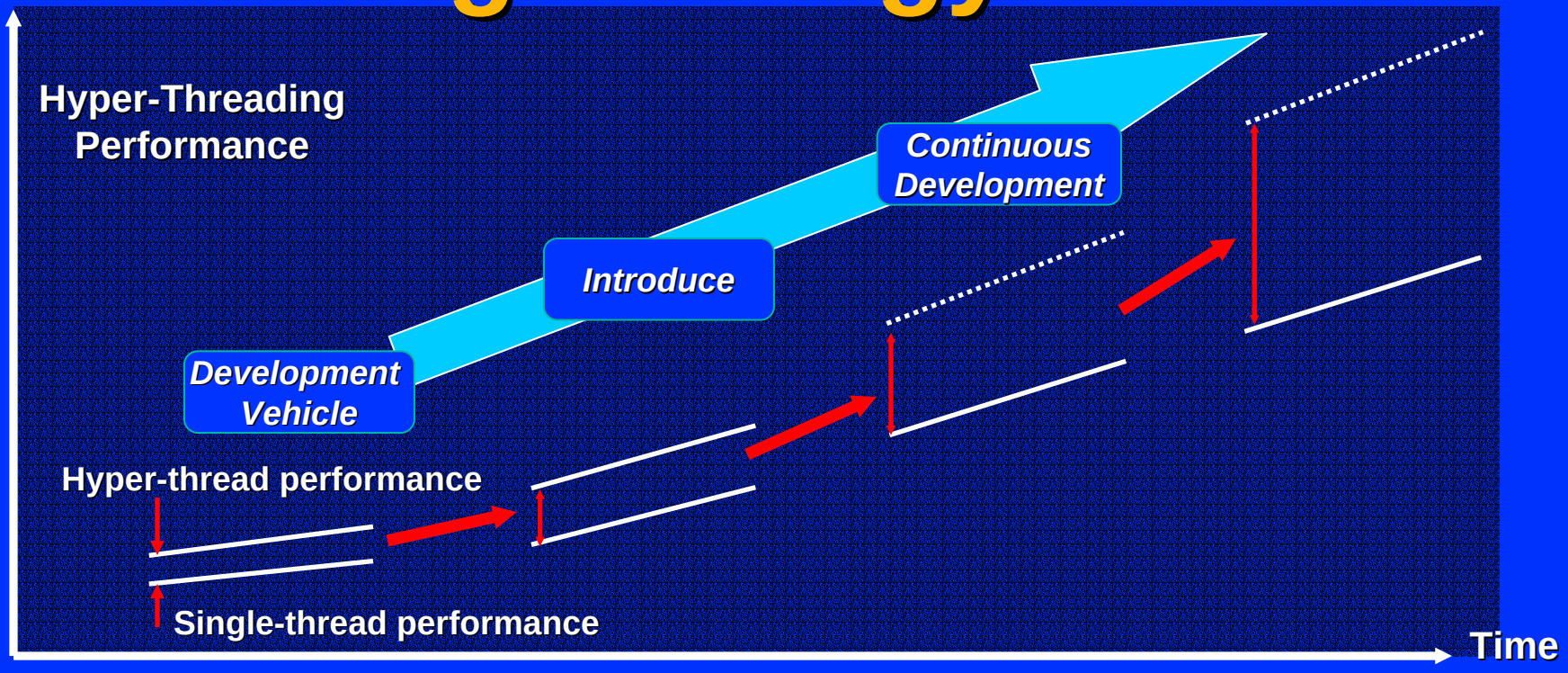
Intel Xeon Processor MP platforms are prototype systems in 2-way configurations
Applications not tuned or optimized for Hyper-Threading Technology

Preliminary Hyper-Threading Technology performance numbers on prototype Intel Xeon processor MP platforms today

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/procs/perf/limits.htm or call (U.S.) 1-800-628-8686 or 1-916-356-3104

Intel Austin

Intel's Long-Term Hyper-Threading Strategy

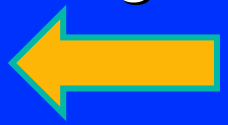


Hyper-Threading Technology expected to deliver increasingly higher performance

Intel Austin

Agenda

- IA-32 Processor Roadmap
- Design Goals
- Architecture 101
  - Frequency
  - Instructions Per Cycle
- Advanced Architectural Concepts
  - Data Speculation
  - Data Prefetching
  - HyperThreading™
- Summary

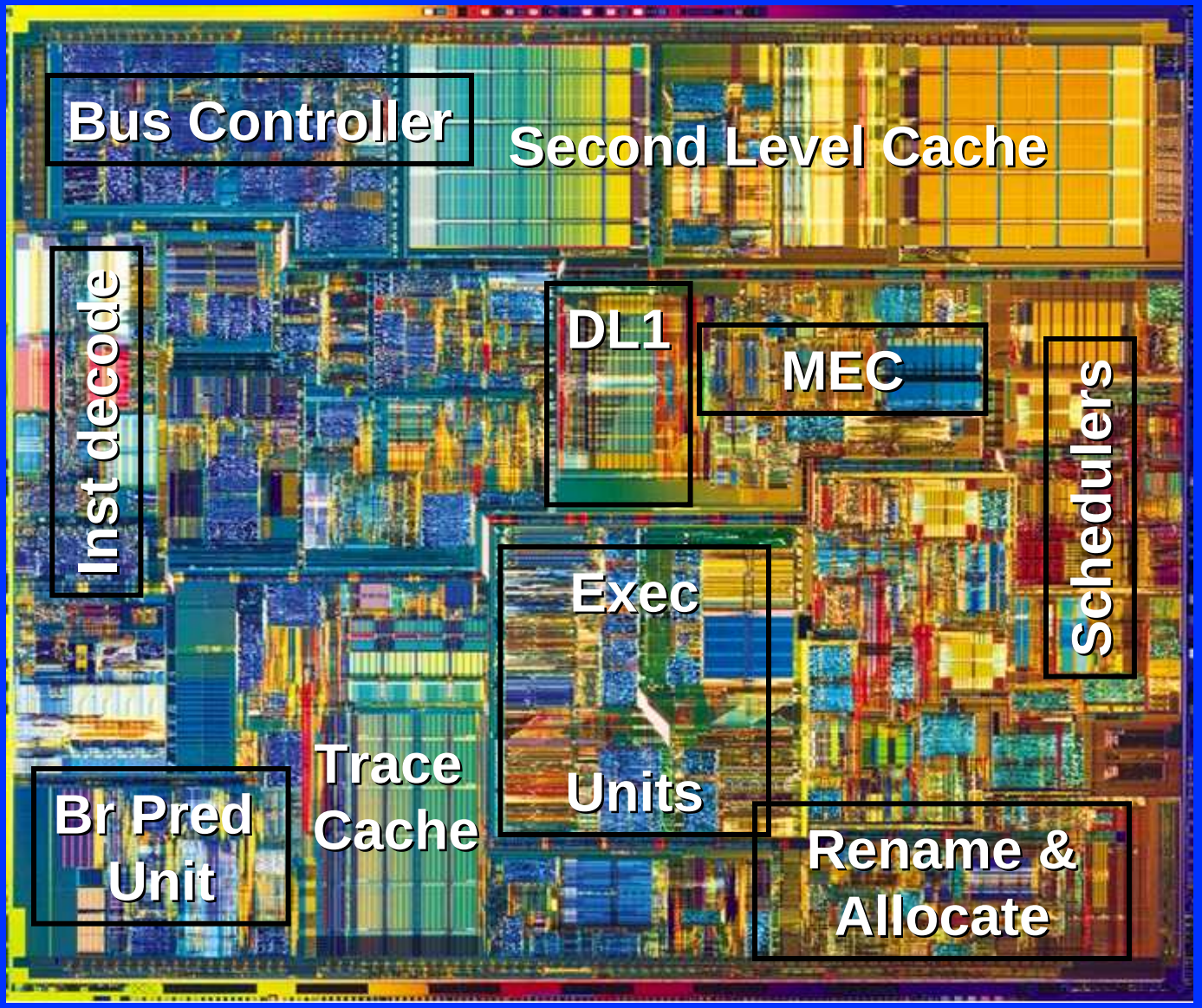


Intel Austin

Intel® Pentium® 4 Processor Summary

- Revolutionary, new micro-architecture from Intel designed for the evolving Internet
- Design features for balanced, high performance platform scalability and headroom
- The world's *highest performance* desktop processor

Intel
Austin



Intel
Austin